程序员的成长课

安晓辉 著



中信出版社·CHINA CITIC PRESS

版权信息

书名:程序员的成长课

作者:安晓辉

中信出版集团制作发行

前言

作为一名普通的程序员：

- 你想买一套房子，不想再租住在远离公司的偏僻地带每天通勤4个小时上下班

- 你想买一部车子，可以周末开着去山里转转，看看红叶听听鸟鸣

- 你想买衣服时去窗明几净微笑服务的商场而不是每次都找一个不知名姓的小二网购经济适用款

- 你想每年出去旅游10次8次，今天在苏梅岛潜水明天在魁北克吃枫糖

- 你想每年给爸爸妈妈5万块的生活费，让他们露出欣慰的笑脸

这些想法不能实现，会经常性地带给你痛苦。这种痛苦，会随着你工作时间的增长而加深，渐渐变成你生活的底色——你的底色原本简单明快，现在幽暗阴郁。

实际上，这些所谓的痛苦，有90%都可以通过钱来解决。前提是：你的价值能够不断提升，赚钱速度超越需求膨胀。消除了这些痛苦，幸福感就有生长的环境，就不那么容易被淹没。

可是作为普通的程序员，你却发现瓶颈一个接一个地扑过来。做技术，不知道怎么做到持续精进、怎么坚持；转管理，又不知如何开

始。结果还没等想明白呢，半载一年就过去了，蓦然回首，好像自己的能力没怎么提高，薪水增速却越来越跑不过通货膨胀了。

有时候你觉得开发工作越来越吃力，转型的呼声越来越高，却不知道如果离开开发岗位自己还能干什么。看着别人可以选择当自由职业者，或者能实现财务自由，内心羡慕，然而转过身却只能叹息：自己的路，究竟在哪里？

仔细想想，你就会发现，要搞定这些事情和问题，只要能赚到更多的钱就可以了！

这个结论很俗吗？

不，现实正是如此！

对于大部分开发者来讲，工作和生活的诸多烦恼，其实都源自于：怎么赚到更多的钱。

要想赚到更多的钱，就要回到问题的原点，想想个人赚钱的本质是什么。

个人赚钱的本质是——**出售时间**！对吗？

从出售时间的角度来看：

个人收入=每天可售时间数量×单位时间价格×单位时间出售次数

在这个公式里，有三个要素，简单描述就是：

- 每天可出售的时间数量
- 单位时间价格

- 同一份时间的出售次数

结合开发者的具体情况，可以找到多种提升收入的方式。参考下表：

时间单价	业余时间工作（时间数量）	一份时间卖多次
提升专项技能	接外包项目	录制技能类课程
提升架构设计能力	技术咨询	个人站点广告位
培养项目管理能力	众包	出版技术图书
提升领导力	技术自媒体	在线技术分享
转管理岗位	撰写技术图书书评	股票、期权
跳槽	翻译外文图书	投资理财
……	……	……

或许你知道所有这些方式甚至知道更多，但是，怎么做到呢？

这是个大问题！

知道和做到之间有一道鸿沟，要想跨越它，你不但要努力，还要讲究方法。

这本书不会承诺“看完本书就能快速赚钱”，不提供任何安慰和幻觉，它只是一本工具书，仅仅是发出一份邀请，让你看到有一些方法可以用来分析自己、帮助自己去有效选择和提升，但最终你是否可以用书里的理念、方法、工具指导自己的工作和生活，则取决于你的认真程度和执行力！

所以，如果你愿意付出努力让改变发生，那么我们现在开始一起行动！

本书共有8章。

第1章，介绍如何选择技术方向，我们结合技术成长三阶段模型，讨论在入行、构建技能树、技术转型、团队技术方案选型等常见场景

中如何选择适合自己的技术，提高增值的效率。

第2章，讲述如何在技术上持续精进，着重讨论如何在工作中将持续提升自我能力和价值落在实处。为此我们引入了个人对标管理法，让你从优秀的同行者、一般性规律、技术自身的深浅层次和软件项目指标四个方面出发，找到随时随地可以引领自己前进的小目标、小台阶，再配合职业目标的指引，小步快跑，日有寸进。

第3章，探讨了开发者无法回避的发展方向——技术管理。从开发者到管理者，不是职位序列自然发展的结果，而是一种转型，且并不适合所有人，因此我们提供了一些工具和方法，让你自测一下，看看自己是否适合。如果你发现自己很想试试管理路线，可以接着看走向技术管理的4种常见方式，然后了解怎样为成为管理者做准备。从概率上讲，成为管理者后，可以通过团队实现更大的产出，也会因此赚到更多的钱，你的未来会有更多可能性。

第4章，讨论了技术管理新人面临的18种常见挑战，比如角色适应、委派任务、激励他人、冲突管理、一对一谈话等。如果你刚刚升任管理者，一定会碰见这些挑战，翻翻本章提到的应对策略，会对你有所帮助。

第5章，介绍了开发者在跳槽时常见的8个问题。比如什么时候跳槽好、依据什么跳槽、选大公司还是小公司、去北上广深等一线城市还是找个小地方享受生活等。这些问题会给你触动，引发你思考自己的选择。

第6章，讨论了如何针对匹配度来优化你的简历。最关键的要点有两个，一是一个岗位一份简历，二是分析目标职位的要求，针对要求呈现你的价值点。你能注意到这两点，就可以大大提升简历的通过率，为自己赢得机会。

第7章，如何在跳槽时获得想要的薪水，这恐怕是每个人都关心的话题。其实从你决定应聘这家公司职位的那一刻就开始涉及这一点了，核心要点是匹配度。所以，从简历优化到了解公司、产品、部门、职位，再到了解目标岗位薪酬区间，你都要围绕匹配度做准备。如何准备？这一章提供了流程、框架和方法。

第8章，转型，也许有一天，我们会离开软件，想要去做别的事情。那么你还适合做什么？想做什么？如果你“拔剑四顾心茫然”，可以看看这里的人、事、物模型和发现职业方向的5步法。只要你做好准备、花些精力，就可以找到适合自己的转型方向。这一章还讨论了转型时如何准备目标职业所需的知识、技能，并提供我和我的朋友们实践过的最佳方法：双职业策略。

在正文之后，有一个附录，是我个人分类总结的书单，对开发者的综合素养提升大有助益。

第1章 如何选择技术方向

努力只有在方向正确时才有价值。

开发者选择技术也是如此。假如你阴阳差错选择了一门过时的技术或者在当地没什么单位使用的技术，你在找工作时就会不断受挫，即便你找到了工作，也会忧心自己的发展前景。

• 你刚进入软件开发领域时，需要选择适合你和市场的技术才能更好地就业。

• 你做了一两年技术工作，开始构建自己的技能树（知识图谱），需要选择与你现有技术相互补充、1+1>2的技术来学习和运用。

• 你的技术能力慢慢在团队里凸显出来，成了先锋官，有新的项目、新的产品时，你总是被赋予探索技术方案的角色，可是一着不慎，你选择的技术就可能导致整个项目的失败。

• 你使用C++语言做了5年开发工作，越来越觉得这门语言不适合自己，想要学习一种新技术作为自己以后的主要开发技术，很担心再选错——因为你做开发的黄金时间，加起来也不过三四个5年。

• 你成为技术负责人或者研发经理，需要决定团队的技术图谱，需要为团队承接的新项目选择技术方案，要考虑哪些因素呢？

所有这些问题，我们都会在本章讨论，我们会找到选择技术方向时要考虑的各种因素，然后结合特定的场景，看看你在选择时需要考虑哪些因素。

通过本章的学习，作为个人，你会掌握怎么选择一个适合自己的技术方向，好的方向会让你越来越有竞争力，越来越值钱；作为团队的一员，你会知道在为某个项目选择技术时该如何综合考虑，在风险、成长、代价之间进行权衡。

1.1 技术成长三阶段模型

为了更好地选择技术方向，我们先来介绍一个模型，我给它取了个名字，叫“技术成长三阶段”，如图1-1所示。

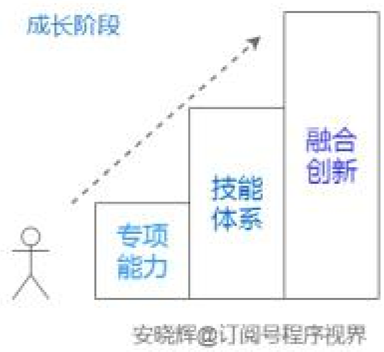


图1-1

我们在一个专业领域内的成长，基本上都会经历三个阶段：

- 专项能力的提升
- 技能体系的构建
- 融合创新

1.初级阶段：专项能力的提升

专项能力提升阶段是初级阶段，你为了做事情，必须先具备某些基础能力，比如从事软件开发工作，你要学会某种语言、某个IDE、某种技术框架。

如果你刚刚从学校毕业开始做开发，或者刚刚从别的跑道转换到开发领域，那么你就处在这个阶段。这个阶段最重要的就是提升专项

能力，让自己能够迅速搞定一些别人安排给你的事情，体现出你的价值。

这个阶段持续的时间长短因人而异，可能会是1~3年。如果超过3年还没有进入下一个阶段，那么你的成长速度就需要提升了。

2.中级阶段：技能体系的构建

这是中级阶段，你拥有了一组技能，围绕某个方向构建了自己的知识图谱，能够用自己的方式来解决问题。比如你使用Java语言做后台方面的开发，你的技能体系可能由Java、MyBatis、Spring、SpringMVC、Netty、MySQL、Python、Linux等组成。此时你在团队中，应该已经可以独立负责某个模块，能够完成模块的设计和开发工作，也能够指导初级阶段的同事进行开发。

你可能需要2~3年，经历2~3个项目才能慢慢构建起自己的技能树，然后还会在这个阶段待上1~2年，不断练习你的技能体系中的各项技能。所以当你能够拥有一棵强大的技能树时，距离你刚从事开发工作可能有5~8年了。

假如你超过这个年限，用过的技术还是散乱的，东一榔头西一棒槌，不能有机组合在一起，那么你的开发经验和能力，肯定大大落后于你的工作年限。换句话说，你可能把一年工作经验学到的知识、技能机械地重复了几年，没有获得应有的成长。

3.融合创新

这是高手阶段，你具有了丰富的实践经验，具备了T型知识结构，形成了自己的思维框架和解决问题的框架，能够融合不同领域的知识，组合各种资源，创造性地解决各种问题。

进入这个阶段非常重要的一种标志就是，你遇到问题，不再从下而上去思考（即从技术实现细节来考虑问题好不好实现、拿技术去裁剪问题或重定义问题），而是从实现细节跳脱出来，站到更高的层面，自顶向下去思考、去分析，先运用框架、逻辑去分析真正的问题是什么、问题的目的、问题的现状、如何去解决。搞明白这些之后，你才会沉降到技术层面去考虑实现的选择，而且实现时，你也不会拘泥于某种技术，而是什么技术合适就用什么技术（你的目的不是“用Java或Redis解决问题”，而是“解决问题”）。简单说，就是你走出了被技术束缚和塑造的过程（前两个阶段），可以反过来回到问题本源来思考了。

在这三个阶段中，我们都会遇到选择技术方向的问题。比如在初级阶段，入门时要选择，发现某种技术不适合自己时要重新选择；在中级阶段，我们要选择某些技术来构造我们的技能树，或者要做技术转型，需要重新选择技术方向；再比如在高级阶段，我们往往会负责产品的技术方案探索与选型，免不了要选择技术方向，甚至在这个时候，我们个人也可能会在技术方向上转型，必须有所选择。

所以，我们根据成长阶段的划分，挑选了下面几个可能需要选择技术方向的时机，分开来讲在这些时机怎么选择技术方向。

- 入行
- 构建技能树
- 技术转型
- 方案选型

但在展开之前，我们需要先通盘梳理一下选择技术方向都需要考虑哪些因素。

1.2 选择技术方向都要考虑哪些因素

我们先来看看选择一种技术可能会考虑的因素，然后看看在每个选择时刻运用哪些因素。

1.就业机会

这个因素，考虑的是哪种技术更容易就业，需要结合特定地区、特定行业来看，因为你找工作是面向地区和行业的。

这中间又有两种典型的策略，热门的和冷门的。比如Java和JavaScript是典型的热门语言，你可以选择学习它们；再比如汇编语言、R语言、Scala语言、Qt框架就没那么热门。

热门和冷门是相对的，根据开发者多少、市场需求而定。

2.自我感觉的难易程度

比如我当年就在Java和C语言当中选择了C语言，因为《The C Programming Language》比《Java 2核心技术》薄得多，面向过程的语言也比面向对象更容易入门（对我来讲）。

难易程度与人相关，所以最好的方法，就是把你感兴趣的语言，都学学试试，可能每种语言花上一到两个星期，你就可以做出判断。

3.兴趣

很多人做开发可能是出于某种兴趣，比如对游戏感兴趣就做游戏开发，对智能手机感兴趣就做APP开发，对电子商务感兴趣就做前端开发。

当你喜欢某一类产品时，这类产品往往可以关联到技术。比如你喜欢做手机游戏，那么可能会选择Unity 3D或者Cocos 2d-x；如果你喜欢樊登读书会这种学习类APP，可能会选择Java、Android、Objective-C、Swift等。

4.薪水高低

大多数人选择技术的出发点是工作和未来发展前景，而选择工作时一定会把薪水作为参考因素，而且是非常重要的因素。所以我们在选择技术时，也可以考虑某种技术对应职位的薪水高低。

要了解这一点，可以去看各种薪酬报告，如100offer每个季度都会有类似的报告出来，拉勾网也会发布互联网职场生态白皮书，你用“互联网人才流动报告”“开发者薪酬报告”或者“互联网薪资调查”为关键字搜索，能搜到很多这类报告。

通过报告，你可以看到使用不同语言的开发者的薪水差异。图1-2是我从100offer 2016年春季互联网高端人才流动报告中截取的。

从图1-2中可以看出不同编程语言所关联职位的平均年薪差异。

5.技术在将来的发展前景

你肯定不愿意看到这种情况发生：选择了一门技术，结果1年后它就没人用、没市场了。所以我们在选择时就要考虑这一点，做一些判断。



图1-2

选择技术时存在两种考虑：一种是选择稳定的、经典的技术，一种是卡位将来的市场缺口，选择将来可能需要用到的技术。

前者考虑的是林迪效应，即：对于不会自然消亡的事物，生命每增加一天，则可能意味着更长的预期寿命。

用林迪效应来看开发技术，一项技术存活的时间越久，其预期剩余寿命就越长。比如C语言（1967年诞生），存活了几十年，可能还会存在几十年。所以你可以选择像C、C++、Java这些经典的编程语言，它们流行了几十年，还将流行几十年。

回顾历史，经典技术都是从新技术发展来的，展望未来，一定也有一些现在的新技术会发展成将来的经典技术。如果你能抓住它，就可以享受它带来的技术红利。

要抓住可能成为经典技术或热门技术的新技术，就要应用**卡位策略**。

卡位策略是指看到某种技术会在未来具有稀缺性，非常有价值，选择提前介入，让自己在未来具备竞争优势。比如2007年Google发布了Android Beta操作系统的SDK，就有人预见到Android开发需求将会爆发，果断学习Android开发；比如现在VR、AR、机器学习、深度学习，其实都还没出现普适性的应用，但是将来一定会出现，此时学习相关技术，将来一定会占据先机；再比如2009年大数据概念兴起，如果那时选择Hadoop、Scala等技术，你现在可能就很值钱了。

运用卡位策略时，有两种方式。

一种卡位方式是根据市场需求和未来预期，就像图1-3所示，左下角是我们现在看到的现状，某些需求处在萌芽期，但是将来可能爆发。这个需求对应的产品、服务可能会用到某种技术，甚至这个需求本身就是由技术驱动的。这时候，你就可以分析从现在抵达将来可能需要用到的技术，提前做准备。

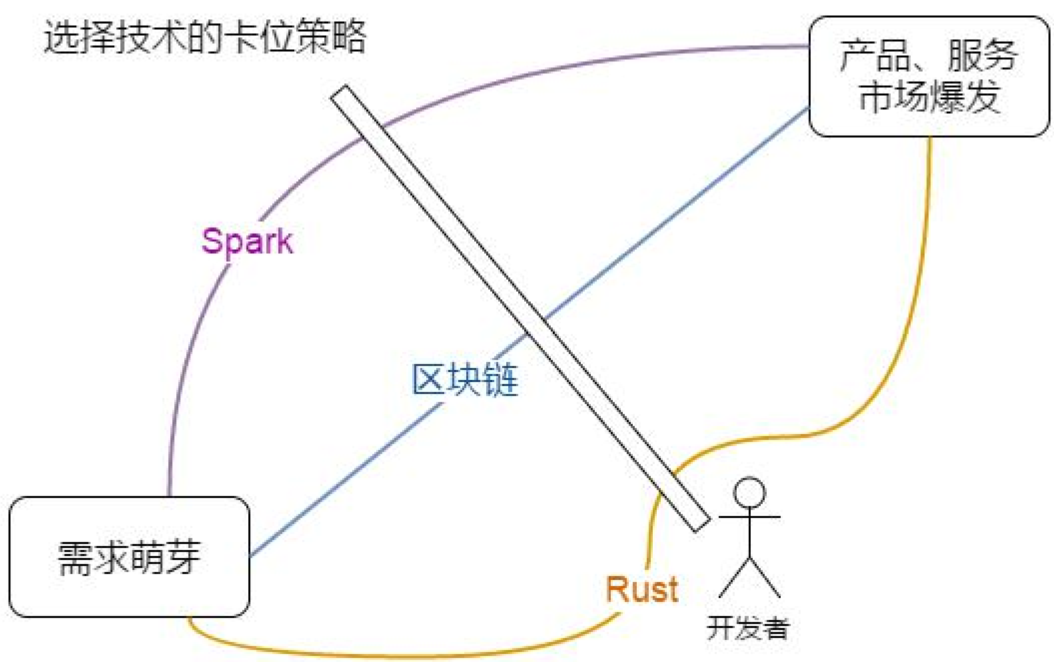


图1-3

还有一种卡位方式，是根据技术本身的发展程度来说看。采用高德纳曲线（参考http://www.gartner.com/technology/research/methodologies/hype-cycle.jsp），如图1-4所示。

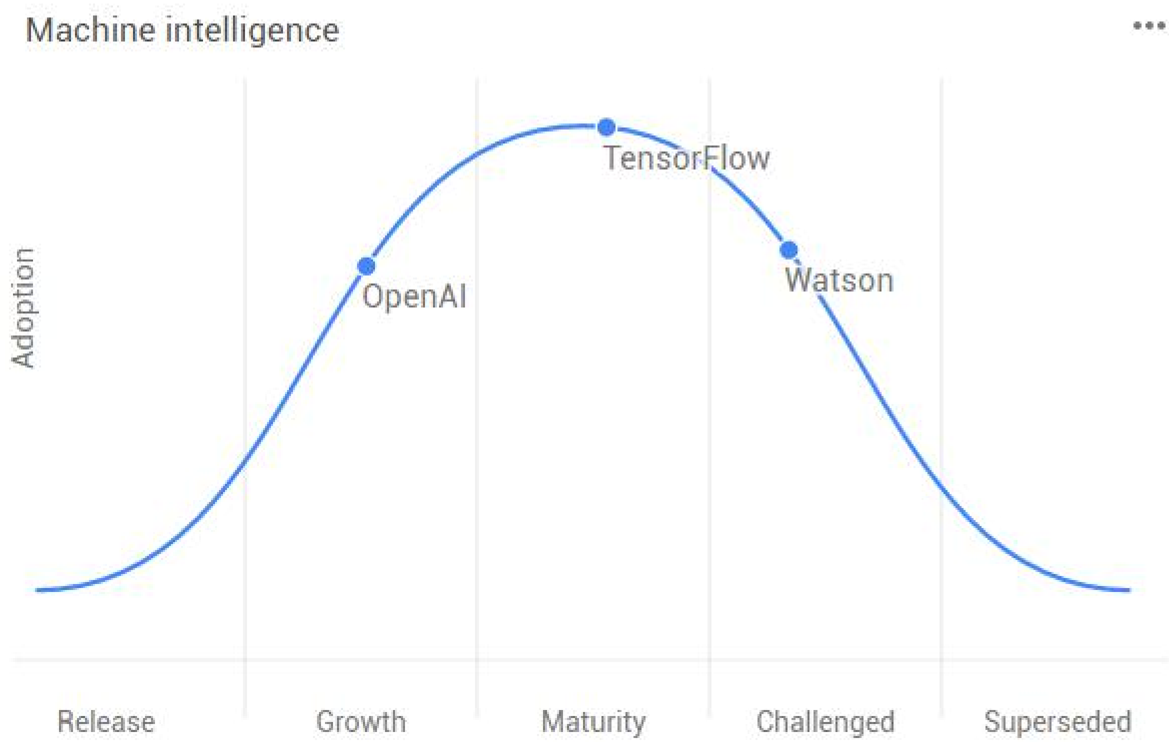


图1-4

图1-4（来自https://stateofdev.com/）展示了机器学习相关的三种技术：OpenAI、TensorFlow、Watson。从曲线可以看出，OpenAI处于成长期，TensorFlow处于成熟期，而Watson的地位正在受到挑战。如果你现在要选择一种机器学习框架，TensorFlow、OpenAI是比较好的选择。

6.你接触的人的推荐

我们是否选择某种技术，和我们看到的可能性有非常大的关系。只有看到它，它才会进入你的选择范围，如果你看不到它，它再牛，

对你也没意义。

比如我2005年自学编程，面临两个选择：C或者Java。我知道Java的存在就是因为读研同学的推荐，而C语言，则是当时大唐电信做开发的同事推荐的。我再没考虑其他编程语言——因为我根本不知道还有什么编程语言。

你看到的可能性，限制或决定你的选择。

7.相近性

当我们已经掌握了一些技术，要学习新技术时，就可以根据一种技术是否和自己已经掌握的技术比较接近来选择。相近的技术，学起来会更容易上手。

8.互补性

和相近性类似，互补性也常用在拓展我们技术能力的情景下。它指的是，有一些技术可以和你已经掌握的技术互相补充，组合在一起，形成更完整、更系统的技术图谱，给你带来更大的竞争力。

9.团队的技术图谱

我们所在开发团队的技术图谱，也可能会影响我们的选择。比如我们基于CEF开发浏览器，团队以C++为主要开发语言，如果自己本来会C++，再学习JavaScript，就可以在JavaScript与C++衔接这部分起到关键作用，从而具有特殊地位。

1.3 入行时怎么选择技术方向

在做选择之前，应该先想想自己的目标，根据目标选择技术，是更理想、更靠谱的。

为了明确自己的求职目标，可以问问自己下面的问题：

- 在哪个城市工作
- 想在哪些行业、领域发展
- 想去什么样的公司
- 想做什么产品

当你能够勾画出工作目标的大概轮廓时，对应的技术方向就会浮现出来。

比如你特别喜欢使用蘑菇街或者美丽说买衣服，尤其喜欢商品展示页面，那么你的目标可以是到美丽联合集团做电商前端开发岗位……此时学前端开发对你来讲可能就是合适的选择。

比如你特别喜欢上海，又喜欢用爱奇艺看视频，对视频类产品很感兴趣，那么你的目标可以是到上海爱奇艺做视频产品开发……此时学习C++、FFmpeg等技术可能有助于你实现目标。

当你有目标时，学习会更有动力，效率也会更高。

一个公司是否选择你，取决于你本身和公司招聘需求的匹配度，而不是某个技术本身是热门还是冷门的。所以在初入行选择技术方向

时，一定要先考虑你将来准备做什么（城市、行业、公司、产品），针对你的目标来选择技术。

当然你可能在想做开发时并没有明确的职业目标（你不孤单，大多数人都是如此），此时你考虑的是这些因素：

- 就业机会
- 薪水高低
- 难易程度
- 他人推荐
- 兴趣

考虑就业机会，有热门、冷门两种策略。热门还是冷门，可以通过各种排行榜看出来，也可以通过在招聘网站挖掘相关职位信息来判断。

选择热门技术，比如Java、JavaScript、C++等，一定有很多机会，看起来你能接触到更多的公司、职位，这会在心理上给初入行的你一种安慰。但并不能说你找到合适工作的概率就很大，因为具体到每个职位，招募方都是选择和自己的需求更匹配的求职者。在这个前提下，因为竞争者众多，且其中有很多有经验的选手，用人单位有更大概率找到比你更匹配他们的需求的开发者，所以你作为初学者可能常常过不了简历关，或者过了简历关又过不了面试关，往往会面临更大的求职压力。

选择冷门技术，机会可能相对较少，但是竞争者也会比较少，而且机会相对确定。比如在西安，如果你以Qt来求职，那么常见的公司可能就有广联达、诺瓦电子、和利时、有智等公司。你可以仔细斟酌

它们的招聘要求，运用心力打造自己的某个亮点，认真地准备简历、笔试、面试等环节，极有可能获得想要的工作。

从心理和统计意义上讲，当你没明确目标时，选择当地需求比较旺盛的技术来学习，是相对稳妥的。至于哪种技术需求量大，到智联、51job、猎聘、拉勾等网站，以地区、技术为条件来搜索职位，看结果多少，就能大概判断出来。

当你这样选择技术方向时，也能够降低他人推荐中携带的个人主观性对你最终选择的影响。

薪水高低这些信息，可以通过招聘网站、行业薪酬报告等来获得，后面我们在介绍工具时会讲到，在“如何在跳槽时获得想要的薪水”那一章会展开介绍。

要想知道自己能不能学会某种技术，最好去试试看、做做看。现在网上有很多免费的学习资源可以利用，我建议初入行的你，根据兴趣和他人的推荐，得到一个技术清单，每一种都摸索、感受一下，然后决定选择哪一种。

对开发者来讲，自主学习能力是非常重要的，这一关一定要过，这也是我建议读者先自己摸索学习的原因。当你自己学到了一些东西，想找人带或者加速的时候，可以寻找私教或者培训结构。

我了解到有这些资源：

• http://tutorialspoint.com，各种入门教程，基本上你能想到的语言和技术框架，这里都有。赞，图文为主，英文版。

• http://www.w3school.com.cn/，网站开发相关技术和教程，非常全。赞。

• 慕课网（http://www.imooc.com），有很多免费的课程，Java、C、PHP、Python、Android都可以找到。

• 网易云课堂（http://study.163.com），有很多免费课程。

• 中国大学MOOC网（http://www.icourse163.org/），也有很多。

• http://codepad.org/，在线代码编辑与调试。

• http://ideone.com/，在线代码编辑与调试。

如果你选择的技术能够和你感兴趣的产品、服务关联起来，那么你学习时会更有动力。所以也请考虑这一点，从你感兴趣的产品、服务出发，找找看它们是使用什么技术实现的，整理出一个清单，帮助你去选择。

1.4 构建技能树时选择技术方向

再来看一下成长阶段模型，如图1-5所示。

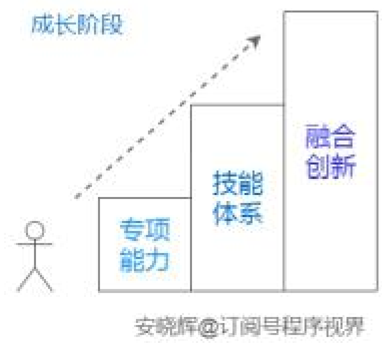


图1-5

当我们过了专项能力提升的初级阶段之后，就应该开始构建自己的技能体系了。在为搭建技能树而选择技术时，通常考虑下面两个原则：

- 相近原则
- 互补原则

1.相近原则

相近原则指和你当前所用语言、技术框架比较接近的其他语言和框架。

比如你使用C语言，可以学习C++、Lua、OC、FFmpeg、OpenGL。比如你使用C++，可以考虑Qt、CEF、Apache C++ Standard Library、Libsourcey、WebRTC、WTL等。比如你用Java开发，可以学习Scala、Groovy/Grails、MyBatis、Hadoop等；比如你用Python，可能对Django或者TensorFlow感兴趣。

2009年到2014年，我个人一直在用C++和Qt Widgets做互联网视频盒子，后来发现Qt里出了新的界面框架Qt Quick，我就去学习，在学习的过程中，我发现Qt Quick内置了Google V8引擎，QML支持嵌入JavaScript语言来脚本化应用，这引起了我的兴趣，就买书学习了JavaScript。在这里，我学习Qt Quick和JavaScript就符合相近原则：Qt Quick和Qt是关联的，JavaScript和QML是关联的。

2.互补原则

互补原则指那些能够组合在一起完成特定任务的技术。比如你学习Scala，可能就会用到Akka；你学习Node.js，可能需要学习MongoDB、AngularJS等。比如你使用PHP，那么可以扩展开来，学习Linux、Apache、MySQL，合起来是LAMP，搭建网站的经典组合。

我们的流媒体系统中有个视频搜索功能，它的实现分为前台和后台。

前台在机顶盒、手机等终端呈现，你输入一个首字母、全拼、混频或者汉字，可以搜索到对应的电影、电视剧、演员等信息。

后台面向前台提供的搜索服务，是用C++结合MySQL实现的。要实现全网范围内的搜索，我们就需要能够跟踪各个视频网站的视频动态，一开始有个同事用C++和HTML来爬取常见的视频网站，后来发现一旦网站结构有变动，修改和验证就特别麻烦，于是就学了Python，用Python来做爬虫，获得原始视频信息，进行初步的结构化处理，然后再入库。

我这位同事学习Python，就是互补原则的应用。这也是我们在做后台开发工作时的常见做法：静态语言做核心业务开发，动态语言负责编译流程、自动化部署、系统资源监控、日志分析、非核心业务处理、测试工具等工作。

1.5 技术转型时的方向选择

做了几年开发工作之后，你很可能会面临技术转型。通常有两种情况：

- 有新技术、新市场出现，你想试试

- 你掌握的技术的应用场景萎缩，前景黯淡，你想跳出来

2017年4月25日，在行约见过我的一位朋友W突然给我打电话，说他要做一个Offer选择，很急。

原来北京有一家在人工智能方向创业的公司愿意给他Offer，要他明天答复。他老婆、孩子在西安，父母也不愿意他抛家舍业去北京。另外现在所服务公司有一个他可以负责新项目的机会。凡此种种，令人纠结，无法决断。

我想起W之前通过在行约我的话题“聊聊程序员职业规划那些事儿”，也是聊如何学习人工智能，意识到这是他真正想要尝试的方向。

在电话里，我问了W一些问题，将他面临的情况简化到3个问题：

- 有多想做人工智能

- 老婆是否真的愿意支持他去北京，关于他到北京工作后，老婆对家里的负担是怎么看的

- 如何在异地经营家庭关系

2017年5月21日我在微信上问W：“北京人工智能那个机会，你后来怎么决定的？”他说：“我已经入职两周了，感觉还不错，每天都在进步。”

这就是一个典型的技术转型案例，W从企业服务外包方向的开发，转向了人工智能，用到的语言、技术框架、SDK、算法等，全都换了。

W这个转型，正是第一种情况：因强烈渴望参与到新技术、新市场当中去而主动转型。这是积极的选择，方向明确，不需要考虑那些选择因素，因为自己的明确目标，已经决定了要用什么技术，所要做的，就是抓紧一切时间去学习目标技术，了解目标市场，寻找机会进入那个市场。

第二种情况的转型，即被迫做技术转型，就需要考虑比较多的因素，与入行时选择技术方向类似。所以，请先回顾一下我们在那里分析的薪水、就业前景（热门/冷门）、兴趣、难易程度、他人推荐等因素。

有经验的开发者做技术转型时，通常有三种情况。

1.在原有技术的基础上做关联转型

比如从Android开发转向使用Java做后台，这是2016年、2017年很多开发者在移动端遇冷时的选择，因为有关联性，转型相对还是比较容易实现的，推荐先在公司内寻找机会。

2.抛弃原来所用技术

比如一直用C#，转换为Go。这相对较难。

我认识一位朋友，做了10年C#相关的开发工作，决定转用其他语言，找工作时，别人总是劝他不要放弃在C#相关技能栈上的积累，劝

他继续用C#做开发，他拒绝了十几家这样的公司，为的就是不再用C#。

在用人单位看来，这位朋友放弃C#是极不明智的，因为他已经做得很好，技术积累的价值都在这个方向上。反过来，我们自己也会这样想问题，你在一个方向沉浸得越久，就越难放弃它去尝试新的方向并以新的技术方向作为自己的主方向。

3.卡位

这个时候卡位市场缺口，学习将来稀缺的新技术是非常不错的选择。而且在这个阶段，也必须积极主动去探索，最好每年给自己定一个学习某种新技术的目标，在工作之外完成它。

要实现这种过渡，一个现实的策略是：一边工作，一边学习。

我个人一直用C++做开发，2013年Android系统在网络机顶盒普及，我们也开始做Android视频盒子。我为了学习Android开发，就给自己安排了一些与Android相关的任务，在工作中完成了对Java语言和Android开发框架的学习。这就是合理利用当下工作中的机会来实践新技术的案例。

你在实际的工作中，可能也存在这样的机会。要勇于去尝试，不要怕多花时间和精力。

如果你在工作中无法应用新技术，可以利用业余时间。比如下班后的几个小时，早起的一两个小时，周末的两天，都可以用来学习新技术。前面我们提到的W为了向人工智能方向转型，买了不少书，还在网上买了很多课程，利用业余时间学习算法、框架，并且自己做一些小练习。

如果你觉得自己很忙，老婆、孩子、父母、朋友都要管，还有各种娱乐也会占用你不少时间，那就测量一下，记录你一周的业余时间都干了什么。数据出来后，你就可以分析，看看哪些事情可以不干，哪些事情可以采用更有效率的方式来干。分析之后，你就能找到时间来学习。

1.6 方案选型

当你构建了自己的知识图谱、能够胜任比较复杂的工作后，在团队中就会越来越重要，就有机会参与到新项目、新产品、新服务的方案选型中来。

做方案选型时，目标是得出现实可行、成本可接受的方案，需要综合考虑各种因素。

1.某种技术适合解决某类问题

很多开发者容易犯“拿着一把锤子看什么都是钉子”的错误，用自己熟练的语言、技术来解决一切问题。比如我喜欢C++语言，就会琢磨怎么用C++语言来做Web服务器，甚至动态Web页面也要用C++来生成；我喜欢Qt，就曾认真考虑过用Qt来做APP，支持Android、iOS、Windows Mobile等平台。

这是典型的被技术塑造、束缚的案例。

实际上，很多技术都有它比较适合的应用场景，比如说Java天生就适合用来做企业应用、服务端应用；C语言离底层就是近，在自动化控制、系统底层、驱动、单片机等领域是无冕之王；R语言被广泛用在大数据分析上……

我们在为新产品、新服务、新项目选择技术方案时，一定要跳出自己已经掌握的技术，首先站在问题域去考虑，想想这个问题到底是什么、哪种技术更适合问题场景，这样才能选择好方向，后期开发、维护才会更顺利。

2.新技术的成熟度

有的开发者喜欢新技术，什么新用什么，把公司的项目、产品当作新技术的试验田，而不考虑试错的代价。这对个人来讲固然可以积累经验，但对公司来讲，往往是一种灾难。

所以，我们在做技术方案选型时，一定要考虑这种技术有没有顶级公司支持、有没有知名产品。假如什么都没有，那么最好先不要用，或者先在非核心业务上应用。

举个例子，很多人都觉得Rust：

- 比Go好
- 比C++好
- 比C好
- 比××好

可是如果让你负责一个要求高并发的服务端应用，你会选择它吗？你能找到一个使用Rust的重量级的后端应用吗？你能找到一个大型软件公司（除Mozilla外）在使用它吗？

3.生态

2014年我负责组建一个团队做互联网彩票，合伙人在邀请我加入前，先找了个小伙伴做后端。这个小伙伴学习能力、编程能力、解决问题的能力都超强，他熟悉Java后端开发、熟悉Scala，熟悉PostgreSQL，还能做Android APP开发，C#也可以。

这个小伙伴为我们的后台选择的技术栈是Scala + Play + Akka + PostgreSQL。Scala发展了几年，也有Twitter这样的公司在使用，应该是比较成熟的。

但当时我们在西安，真的很难找到一个了解Scala语言的开发者！我花了几个月时间才联系到一个懂Scala的小伙伴，可他还不愿意来。最后我只好自己学、安排公司的其他小伙伴学。

这就是最大的问题——没有一个良好的生态。此时

- 找不到合适的人来做开发
- 找不到合适的框架来使用
- 遇到了问题找不到交流的人

非常痛苦。

4.团队的技术图谱

我们做方案选型时，也要考虑团队已有的技术积累。

比如你的团队原本是做机顶盒的，主要使用C、C++和汇编语言，现在因为卖产品的缘故，需要做一个简单的电商的后台，由你负责。

你会选用什么技术来做？C、C++、Python、Ruby、Java、Scala还是Go、Rust？

你会招募一个小伙伴，还是要你现有的成员学习后端开发？

假定你想让现有的成员学习电商网站开发（很多中小公司都会这么搞），那么选择什么技术栈？Python + Django还是Ruby on Rails？

如果我是负责人，恐怕会选择Python，因为它与C语言更亲近，而团队里的小伙伴都熟悉C和C++。

5.技术引入的成本

引入一种新技术的成本，包括：

- 学习成本
- 招聘成本
- 时间成本
- 机会成本

学习成本是引入任何新技术都必须要负担的，就看新技术对你的团队成员来讲学习难度如何。前面讨论团队技术图谱因素时引入的例子——嵌入式团队做电商站点，使用Python + Django，学习成本可能会较低；使用Java + Spring + SpringMVC + MyBatis，学习成本就会高很多。

考虑学习成本，要结合现有技术积累来分析。

为了新技术要招募新的开发人员时，就会产生招聘成本，招聘的成本包括笔试、面试、时间、新成员与团队成员磨合等，成本非常高。

时间成本主要是指引入一种技术，从学习到产出结果之间的时间周期。这个时间周期分为两部分：验证周期和产品化周期。

所谓验证周期，指你通过了解该技术的适用场景、测试关键技术特性、构建DEMO等手段来验证该技术对你的产品的可行性所花费的时间长度。这个周期，从几周到几个月不等。

所谓产品化周期，指过了验证周期，使用该技术把产品做出来的时间长度。可能是从零开始做，也可能是在验证周期内搭建的DEMO的基础上来做。

如果验证周期内你做了一个产品DEMO，简化实现了产品的大部分功能，项目的相关干系人就会大概率地错误评估产品化周期——他们会认为DEMO已经出来，产品化指日可待。也许只要一周、两周、最多一个月，这就是他们的想法。而实际上，你的DEMO仅仅是最小化验证可行性，简化了功能，忽略了异常，各种功能之间也没有按产品化所需的逻辑链接起来，总之它离一个产品还有十万八千里，你要想产品化，其实和没有DEMO时花费的时间差不多——从零开始需要10个人月，有了DEMO，可能需要9.5个人月，事实就是如此。

机会成本有两层含义：

- 你可能选错了技术，导致后期开发、维护成本高昂；
- 很可能另一种技术更适合你的产品或服务，而你已经在不那么适合的技术上投入了太多，改弦更张的代价太高，导致你很难回头。

举个例子，2013年，你们公司要做一个Web IM产品，支持文字、图片，声音、视频等形式的聊天，希望它能在各种浏览器上都可以使用，能无缝集成到客户的电商网站上。

考虑到各种浏览器的历史版本兼容问题，你选择了使用Flash来实现，经过测试，在IE、FireFox、Chrome、Opera、360安全浏览器、搜狗、QQ浏览器、猎豹等浏览器上都可以正常使用，考虑到电商网站主要通过桌面计算机访问，所以最终你们采用了Flash的方案。

可是后来移动互联网快速发展，iPhone、iPad用户很多，默认浏览器是Safari；Android手机星火燎原，内置浏览器也不支持Flash。使用这些设备的用户无法使用你们的IM产品，导致集成你们IM产品的电商公司不断投诉你们。这时你才发现，再想换到WebRTC或者其他技术框架，已经很难了——因为你们前端用Flash技术，后端用Wowza，音视频传输用RTMP，和Flash技术栈深度绑定了。

1.7 工具推荐

这里推荐一些工具或者方法，你可以使用它们来判断某种技术的状况，决定自己是否选择。

1.Stack Overflow

Stack Overflow的开发者调查：http://stackoverflow.com/insights/survey/2016。更改最后的年份，比如将2016年修改为2015年，即可查看对应年份的调查结果。这个结果里有开发者最喜欢的语言、最想要学习的语言、开发者年龄分布、Stack Overflow上的热门技术等，很有参考意义。

比如2016年最流行的技术如图1-6所示。

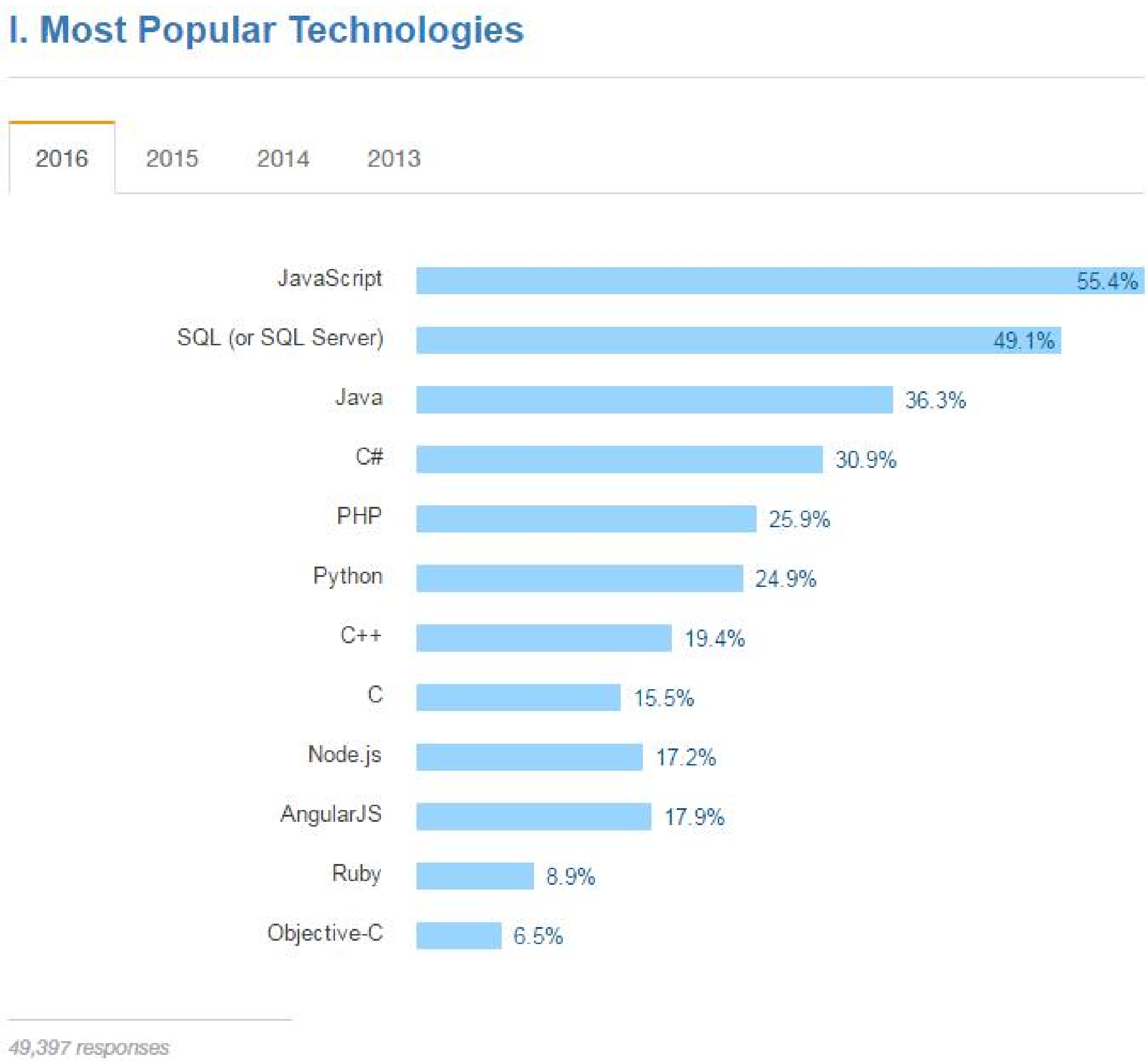


图1-6 （来自Stack Overflow）

哎，JavaScript，前端还是热啊。

比如2016年Stack Overflow上的热门技术，如图1-7所示。

响很大，仅供参考。比如Rust最被热爱，并不代表它未来就可以获得

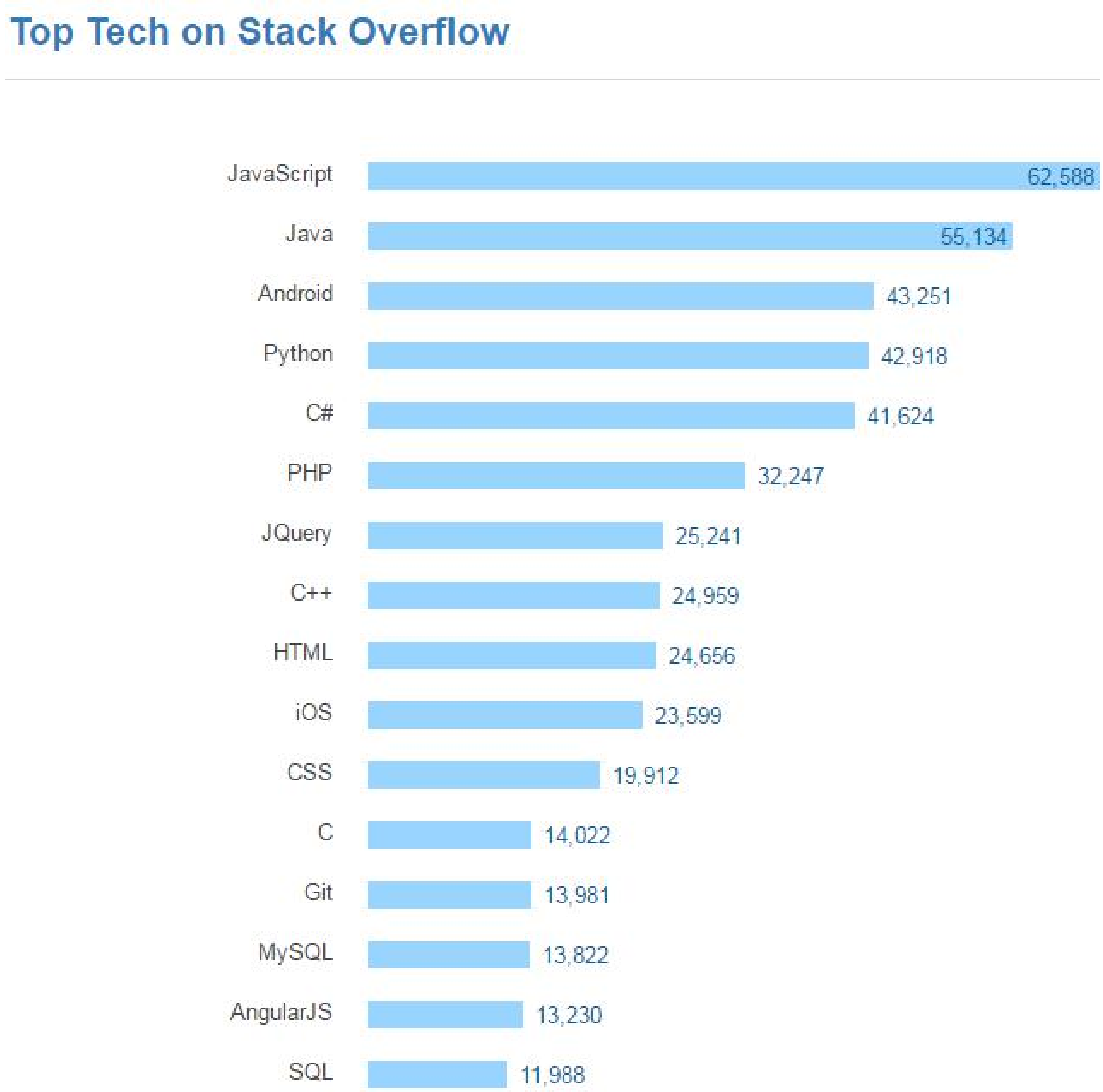


图1-7 （来自Stack Overflow）

哇，JavaScript、JavaScript……

比如最被热爱的技术，如图1-8所示。

需要提请注意，Stack Overflow的统计结果受样本范围和数量影响很大，仅供参考。比如Rust最被热爱，并不代表它未来就可以获得

大面积应用，也不代表它的生态很快就能完善，很快就会出现杀手级应用。

2.TIOBE编程语言排行

TIOBE编程语言排行：https://www.tiobe.com/tiobe-index/。会有TOP 20编程语言及TOP 10的走势曲线。比如现在TOP 20语言，如图1-9所示。

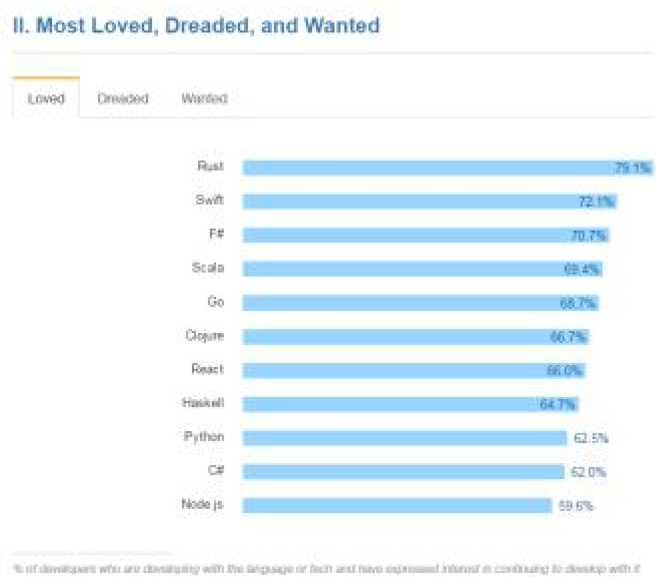


图1-8 （来自Stack Overflow）

Mar 2017	Mar 2016	Change	Programming Language	Ratings	Change
1	1		Java	16.384%	-4.14%
2	2		C	7.742%	-6.86%
3	3		C++	5.184%	-1.54%
4	4		C#	4.409%	+0.14%
5	5		Python	3.919%	-0.34%
6	7	↑	Visual Basic .NET	3.174%	+0.61%
7	6	↓	PHP	3.009%	+0.24%
8	8		JavaScript	2.667%	+0.33%
9	11	↑	Delphi/Object Pascal	2.544%	+0.54%
10	14	↑↑	Swift	2.268%	+0.68%
11	9	↓	Perl	2.261%	+0.01%
12	10	↓	Ruby	2.254%	+0.02%
13	12	↓	Assembly language	2.232%	+0.39%
14	16	↑	R	2.016%	+0.73%
15	13	↓	Visual Basic	2.008%	+0.33%
16	15	↓	Objective-C	1.997%	+0.54%
17	48	↑↑	Go	1.982%	+1.78%
18	18		MATLAB	1.854%	+0.66%
19	19		PL/SQL	1.672%	+0.48%
20	26	↑↑	Scratch	1.472%	+0.70%

图1-9 （来自TIOBE网站）

看到了吧，很多人黑C#、黑C、黑C++，但是它们依然排在前几位！有时小圈子的热烈讨论会遮蔽事实的真相——新生的、非主流地位的语言更容易引起开发者的讨论，这些频繁的讨论，会让人误以为这些新语言很热、很火。

再看看TOP 10语言多年的走势曲线，如图1-10所示。

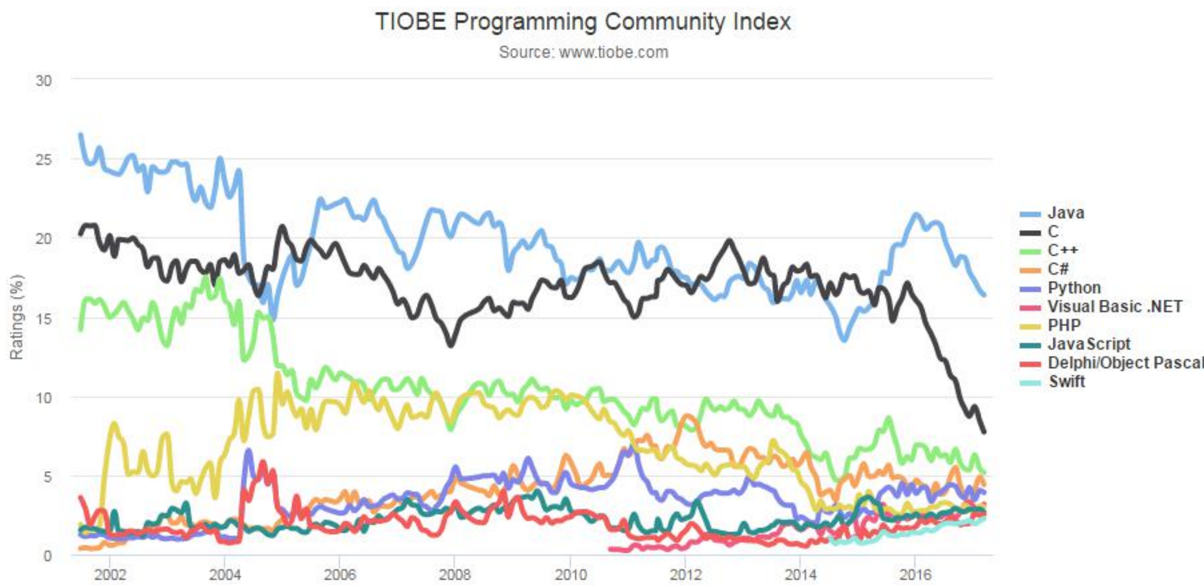


图1-10 （来自TIOBE网站）

3.技术热门度曲线

全球最大的IT咨询公司高德纳提出了一个“技术热门度曲线”模型，把一门技术的发展过程分为启动期、泡沫期、低谷期、爬升期和高原期5个阶段。https://stateofdev.com/这个网站根据高德纳的模型，做了各种开发技术的曲线（部分曲线的阶段划分和描述与高德纳的技术热门度曲线模型有所区别）。

机器学习框架的曲线，如图1-11所示。

可以看到，TensorFlow（Google）已经处于成熟期，OpenAI（诸多硅谷大亨联合建立，开源非盈利组织）在成长期，而Watson（IBM），正在被挑战，快要进入衰退期。那么如果我们要做出选择的话，可能会优先选择TensorFlow或者OpenAI来学习。

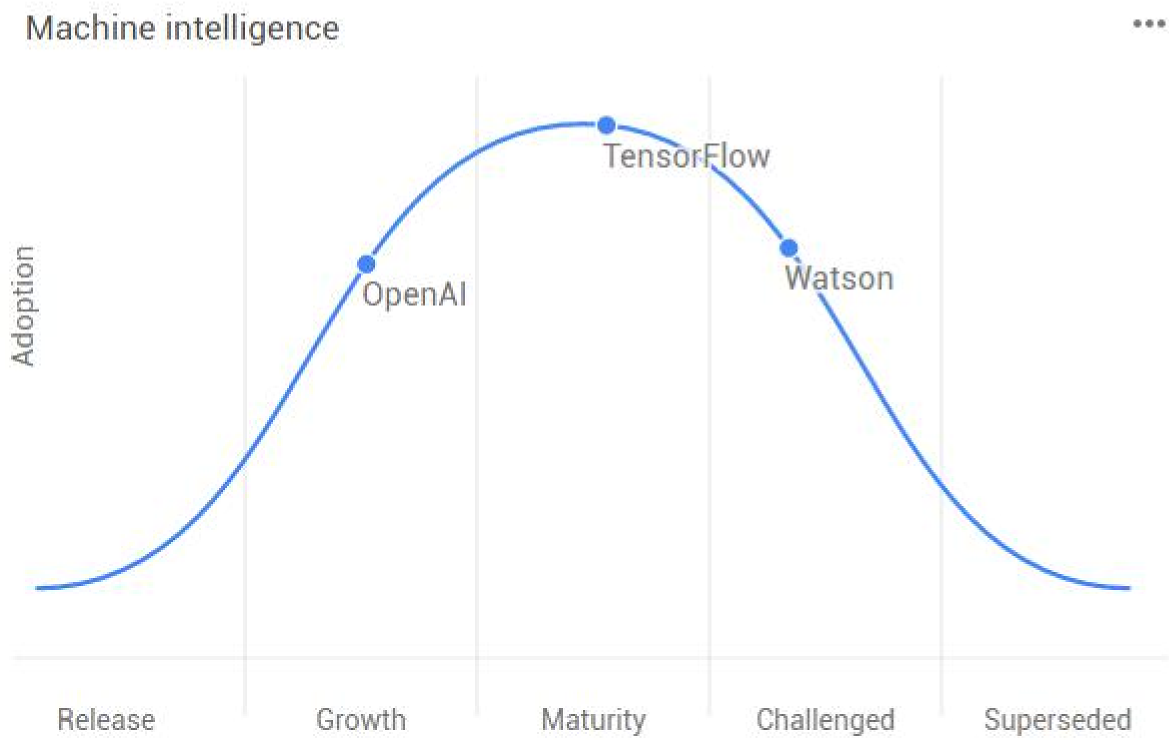


图1-11 （来自stateofdev网站）

后端开发语言的技术热门度曲线，如图1-12所示。

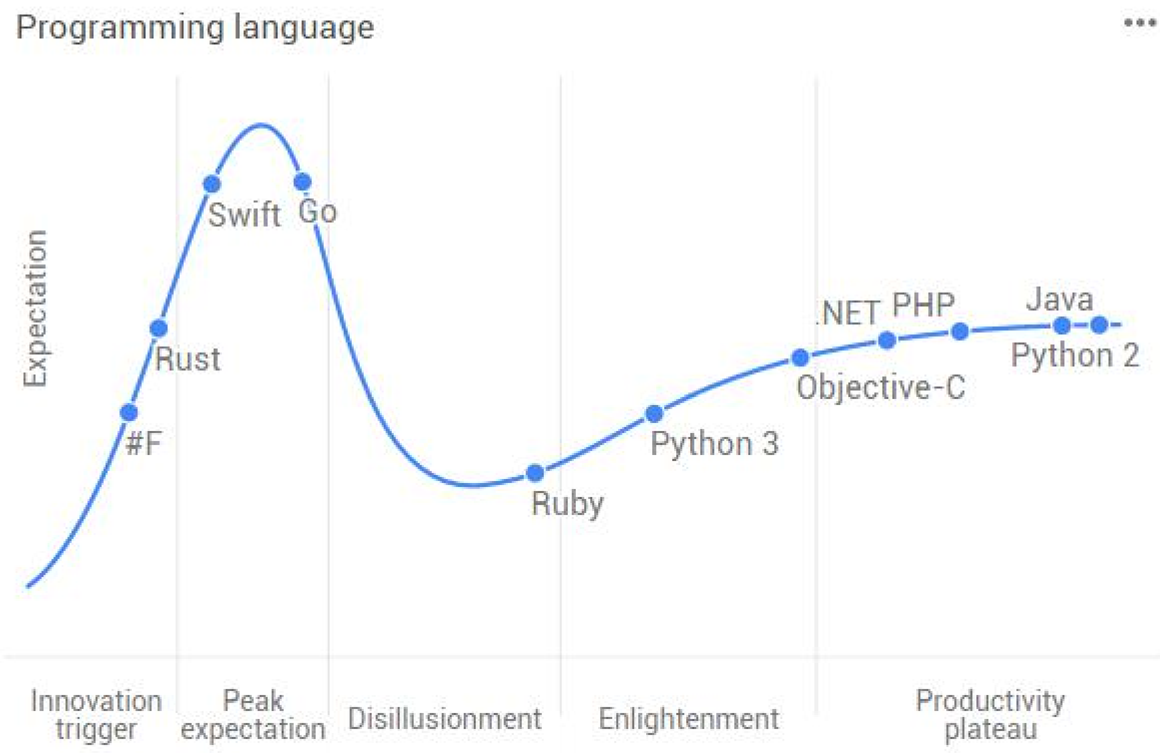


图1-12 （来自stateofdev网站）

前端框架的发展阶段曲线，如图1-13所示。

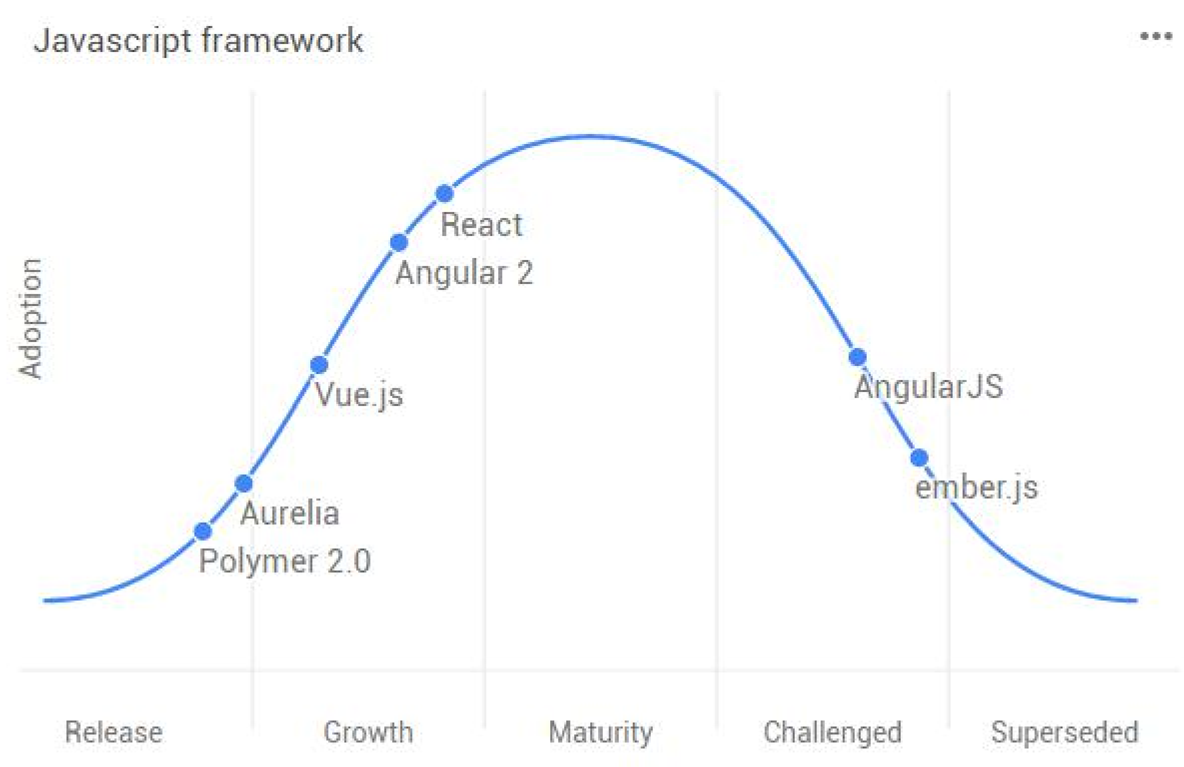


图1-13 （来自stateofdev网站）

4. GitHub

GitHub也有一些统计数据，说明各种语言、技术在GitHub上的活跃程度，可以登录这里查看：https://octoverse.github.com/。另外，你也可以自己在GitHub上按关键字（比如Qt）来搜索，看某种技术是否有很多的项目，以此来判断技术的活跃度，如图1-14所示。

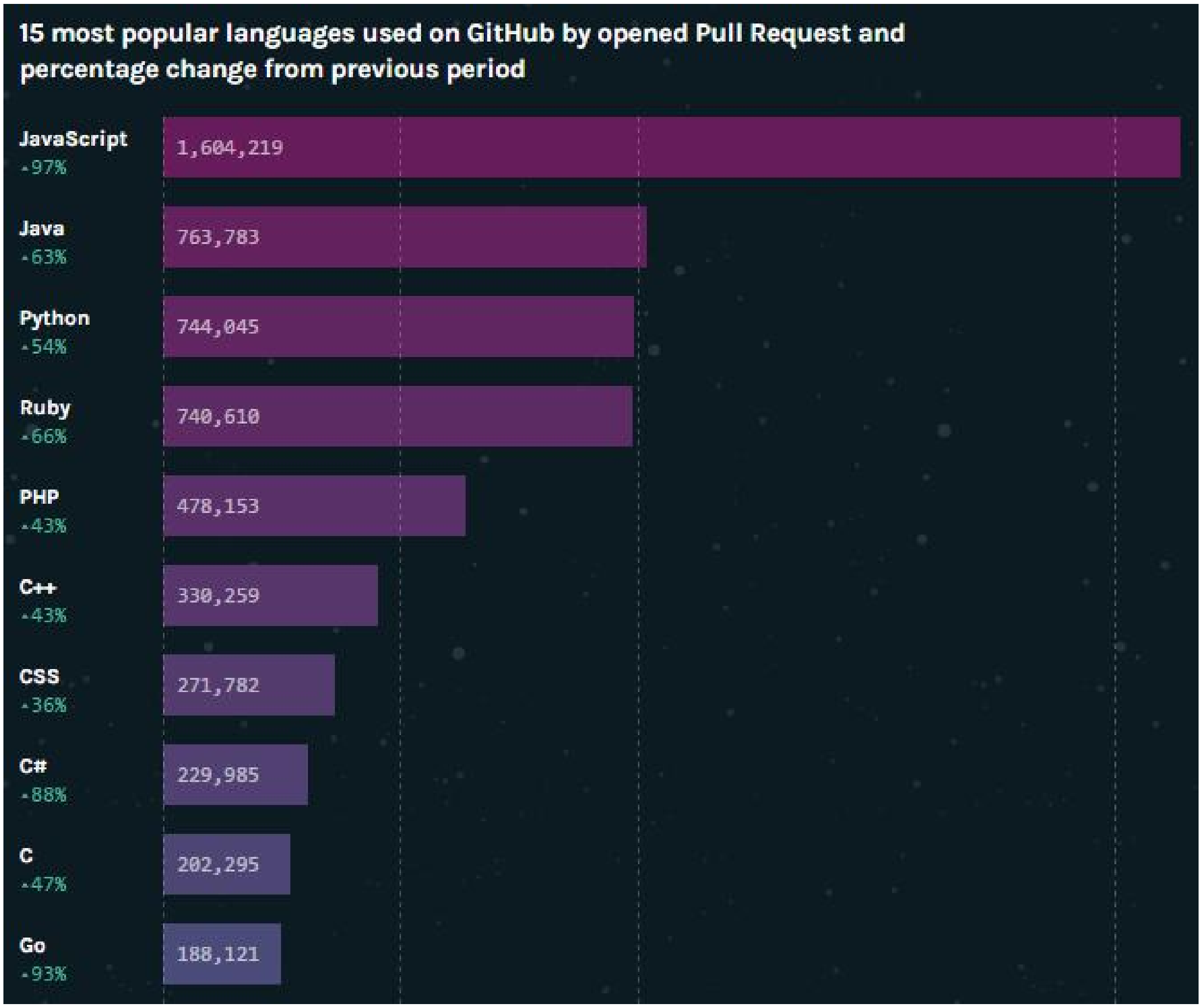


图1-14 （来自GitHub）

5.招聘热度挖掘

像智联、51job、猎聘、拉勾、100offer、大街网等招聘网站，都是很好的技术需求数据来源，你可以用某个关键字（比如“Java开发工程师”）来搜索，看相应的招聘需求是否旺盛。

6.薪酬报告

以“互联网人才流动报告”“开发者薪酬报告”“互联网薪资调查”“程序员薪酬报告”“互联网薪酬报告”等为关键字在百度搜

索，就可以搜到拉勾、智联等网站发布的互联网人才薪酬报告，可以根据这些内容了解某些岗位的薪资状况。

如果你想了解特定地区、特定岗位的薪水状况，也可以到猎聘、拉勾等网站上去，按地区搜索目标岗位，很多岗位都会给出薪水范围，多看一看，就能了解到平均水准。

图1-15所示是我在猎聘网上搜索西安地区的C++开发岗位得到的结果。



图1-15

图1-16所示是我在拉勾网上搜索西安高新区、有3~5年经验的C++开发工程师得到的结果。

可以看到，现在很多公司在招募人才时都会提供薪水范围，这是因为它可以大大提升匹配成功率，节省公司和应聘者双方的成本。



图1-16

第2章 如何在技术上持续精进

作为开发者，工作2~3年后，往往会感到迷茫：能够使用一种或者几种技术解决一些问题了，却觉得自己停滞了，不知道接下来怎么办。于是我们有时羡慕做管理的，觉得他们轻松，地位高，赚钱多；有时又羡慕技术大咖，觉得他们能搞定别人搞不定的问题，像神一样会发光，可是回看我们自己，到底是接着做技术还是转管理呢？如果选择做技术，接下来该怎么精进？如果不做，又该怎么转型？这都是问题，每日煎熬着我们。

我有十几年的开发和管理经验，对于开发者的各种迷茫，深有体会。一般来看，开发者的迷茫分两个层面：

- 方向上的迷茫，即自己到底是否适合做开发、是否要继续在开发的路上走下去。

- 执行层的迷惑，即自己继续做开发，该怎么找目标、学点什么、学到什么程度、如何能持续精进。

本章就从这两方面来展开，先讲如何确认自己要继续做开发还是转管理，这部分会提供一些方法（比如工作的三种分类、成就感来源等），让大家能够自我觉知和判断；然后我们会沿着开发这条线继续前进，看看如何在技术上持续精进，这部分最重要的是对标管理法及其四种典型标杆；再接下来会讲怎样设定有效目标，找到下一步行动；最后，我们会介绍四个习惯，让精进成为你自身的一部分。

先给出本章的思维导图，方便对照阅读和快速抓取要点，如图2-1所示。

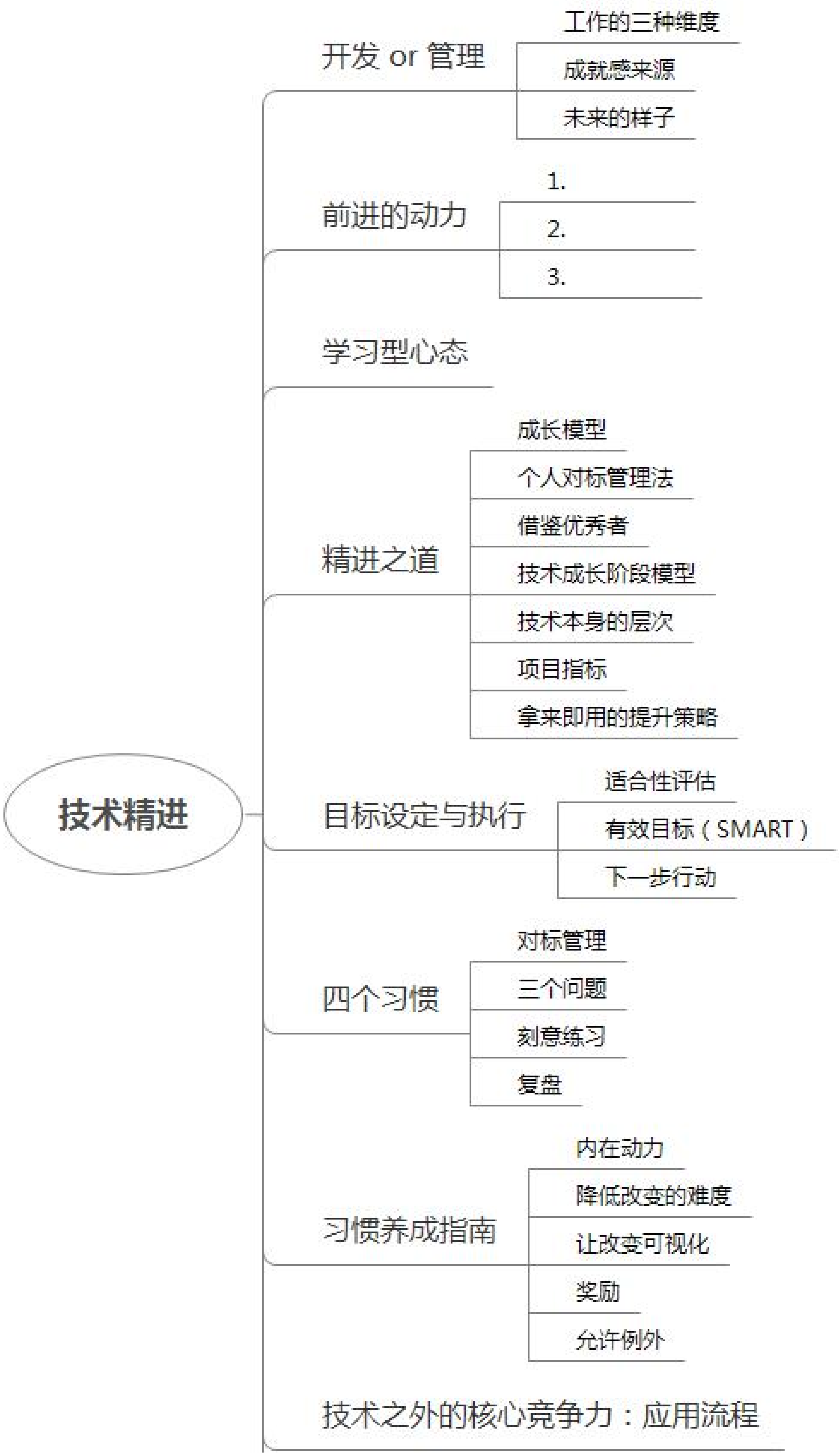

技术精进
开发 or 管理
工作的三种维度
成就感来源
未来的样子
前进的动力
1.
2.
3.
学习型心态
精进之道
成长模型
个人对标管理法
借鉴优秀者
技术成长阶段模型
技术本身的层次
项目指标
拿来即用的提升策略
目标设定与执行
适合性评估
有效目标（SMART）
下一步行动
四个习惯
对标管理
三个问题
刻意练习
复盘
习惯养成指南
内在动力
降低改变的难度
让改变可视化
奖励
允许例外
技术之外的核心竞争力：应用流程



图2-1

2.1 做开发还是转管理

我们的假设是：你已经走在软件开发的道路上了，做了一段时间软件开发，1~2年，或者3~5年，对接下来该继续做开发还是转向技术管理产生了困惑。

在此前提之下，有三种方法可以帮助你进行判断：

- 工作的三种维度；
- 成就感来源；
- 未来的样子。

1.工作的三种维度

根据交互对象不同，工作可分为三类：

- 数据和信息处理；
- 人际互动；
- 事务型操作。

假如你发现自己更愿意围绕着人际交互来做事情，希望自己的工作中大部分时间都在和人打交道，那么你可能更适合做管理、销售、市场、客服、咨询师等方面的工作。

假如你发现自己更愿意做操作性的工作，比如修理电脑、组装电脑、搭建局域网、修理汽车等，那么软件开发工作可能不太适合你，运维或者网管也许和你更匹配。

假如你觉得信息很迷人，很享受与信息和数据之间这种确定性、一致性、可预期性较高的互动方式，也很享受通过组织、修改、整合、创造信息来解决问题这种工作方式，那么你现在正在做的开发工作，基本上和你的偏好是吻合的——因为开发者偏重于和数据、信息打交道，以信息和数据为输入，也以信息和数据为输出。

2.成就感来源

2015年年底创业失败，我决定找一家单位上班。彼时我35岁，在很多人眼里这个年龄的程序员已经要被淘汰了。我面临的选择是：做开发还是做管理。

从2009年开始我就在做技术管理工作，这时比较传统的做法是，找一个研发经理的职位来做（当时有几个不错的机会），一来职业生涯有延续性，二来薪水也高。可是我后来选择了到全时云商务做开发工作，让很多朋友大跌眼镜。

我为什么这么选择？其中的关键点是：我觉得亲力亲为解决问题更有成就感。

回顾我多年的开发和管理经历，我发现在我写作《Qt Quick核心编程》时，在我一周7天不休息，加班加点重构智能机顶盒播放器时，感觉最为充实，最有意义。而在我做管理工作时，带领团队完成了某项任务，自己也没什么特别的感觉，即便有一些兴奋感和成就感，也很快会被委派任务、一对一谈话等事情淹没。

所以我思来想去，决定做回开发工作，这样我更能感受到意义和价值，更有成就感。

每个人的成就感来源都不一样，假如你像我一样，成就感在于自己动手解决具体的技术问题，那么开发工作就更适合你；假如“领导

和管理别人、通过别人完成工作、看到别人成长”让你更有成就感，那么管理工作适合你。

实际上这一点和前面介绍的“工作的三种分类”是类似的。你越倾向于做人际互动类的事情，就越适合做管理工作，你越倾向于和数据、信息交互，就越适合做开发工作。

可以遵循下面的步骤寻找成就感来源：

• 回顾你做过的事情，找出那些让你情感反应强烈的，记录下来。

• 分析你的情绪底色，是快乐、高兴、振奋、愉悦、充实等积极情绪，还是沮丧、灰心、挫败、失落、空虚、失望等消极情绪。

• 挑选出带给你强烈积极感受的事件，它们就是你的成就感事件。

• 分析成就感事件，看看它们用到了什么知识、技能、软能力，看看在这些事件中，你印象深刻的交互对象是什么（数据、人、事务）。

可能有的朋友会说，即便有这4个步骤，也判断不出自己的成就感来自哪里，不怕，第8章——转型——还有更简洁的方法可以帮到你。

3.未来的样子

问自己两个问题：

• 一直做开发，我会变成什么样子？

• 如果转向管理，我会变成什么样子？

在你的身边找比你大5岁、8岁、10岁的开发者，看看他们的工作状态、生活状态、个人风貌，是不是你想要的，或者反过来，是不是你讨厌的。

假如看到他们你觉得看到了自己的未来，这个未来你无论如何无法接受，那么，很可能软件开发并不是你的长久之道，你可能要考虑技术管理岗位。

那就在你的身边找管理者，看看直接上级，看看其他部门的经理，看看其他公司的管理者，找5个以上的人出来，观察、了解、搜集他们的工作状态、生活状态、个人风貌等方面的信息，看看这样的人是不是你想成为的，或者反过来，是不是你讨厌的。

假如你也很讨厌你能接触到的这些技术管理者，那么，可能这条路也不是你的长久之道。你的未来应该在别处。你可以考虑转型，那么请去看第8章的内容。

2.2 找到激励你前进的动力

你之所以愿意做软件开发，一定是有原因和目标的。比如：

- 它可以带给你丰厚的薪水和优裕的生活；
- 它能让你成为自己想要的样子；
- 它可以不断给你挑战；
- 它让你有成就感；
- 它能让你获得认可；
- 它让你觉得自己有价值。

……

请问一问自己为什么要做软件开发，找到几个理由，记录下来。

它们会帮助你渡过后面会遇到的倦怠期。

进入软件开发领域后，你从对什么都好奇、积极学习，慢慢变成机械地完成领导分配的任务，3~5年后就可能会感到无聊、困惑、迷茫，丧失前进的动力，进入倦怠期。

这个时候，你对工作的态度是“差不多就算了”，干什么都没心思，开始思考朝九晚五打卡上班有何意义，整个人陷入“有事不想做、没事又无聊”的倦怠之中，更谈不上什么技术精进了。

这是一种危险的状态，要让自己走出来，有两个方向：

- 为你的工作重新赋予意义。
- 为你的工作引入变化。

具体可以像下面这么做，想想你之前记录的做开发的理由，问自己几个问题：

- 这个理由是否还存在？
- 为什么这个理由还在但是无法激励我了？是不够量化，不够可视化，还是太遥远？
- 有哪些新的工作内容可以去尝试？
- 有哪些新的职位可以去尝试？
- 有哪些新的方法、工具可以提升开发效率？
- 当下，软件开发对我的意义在哪里？
- 三五年后，我会变成什么样子？
- 我希望自己变成什么样子？
- 该怎样才能变成我想要的样子？

做软件讲究迭代，讲究热更新，我们个人也是一样的，要不断自我探索和更新，找到技术在不同阶段对自己的不同意义，只有找到开发工作对个人的意义所在，你才可能有动力去精进。如果你觉得做不做开发、做好做坏无所谓，那么你铁定是无法持续精进的。

2.3 学习型心态

所谓学习型心态，指的是：有主动学习的意识，时刻以学习的眼光和心态来看待发生在自己身上的事情。

举个例子，作为开发人员，你做完了一个功能，展示给领导看，结果他说你做出来的东西根本不是他们想要的，简直一无是处。这个时候，你会怎么办？

一个选择是怼回去，大家吵一架，然后不欢而散，甚至因此和领导不睦，最终觉得领导太没水平，此地无法发展，离职。

另一个选择是坐下来分析：

- 哪里不对？该怎样调整才符合需求？
- 这个“不对”是在何时被引入的？
- 以后如何避免开发过程中的走样？
- 沟通机制是否存在问题？如何改进？

后面这个选择就是积极主动的选择，是面向解决问题的，是学习型心态的开发者会采取的方式。他们会考虑自己怎样去改进、怎样能达成目标、通过这件事自己能收获什么。

作为软件开发人员，其实有很多提升自己的机会和方式，而最终你能不能提升，除了要看你是否有内在动力，还要看你是否拥有学习型心态。

2.4 技术精进之道

做了各种铺垫，我们终于可以开始介绍精进的方法了。

对标管理法

在专业领域成长的一般模型如图2-2所示。



图2-2

模型中有三个要素：

- 现状
- 目标
- 执行计划

每个人都可以评估自己的现状，自己在做什么、用什么技术、技术达到了什么程度、拿多少薪水、是什么职级、是否被领导认可、与人协作是否顺畅……有很多维度，静下心来思考一下，在纸上列一列，就能得出自己当下的状态。

而目标则很可能随着旧目标的达成而消失，或者随着日复一日的编码、Debug、交付而褪色，或者随着每个月的薪水蒸发掉。一旦我们失去目标，就会陷入迷茫，被动工作，进而慢慢失去竞争力。

所以，要想日有寸进，必须要在日常的开发工作中找到努力的目标。这非常关键——很多人就是因为没有目标而放任自己随波逐流、被动工作，最终变得庸常而被组织淘汰。

因此我们引入原本用于企业的对标管理法，帮助自己在日常工作中找到贴合自己的目标。一旦我们找到目标，对比现状，就可以找到差距和前进方向，有了方向，就可以制定计划，稳步前进，获得提升。

图2-3是实践对标管理法指导个人成长的基本过程。

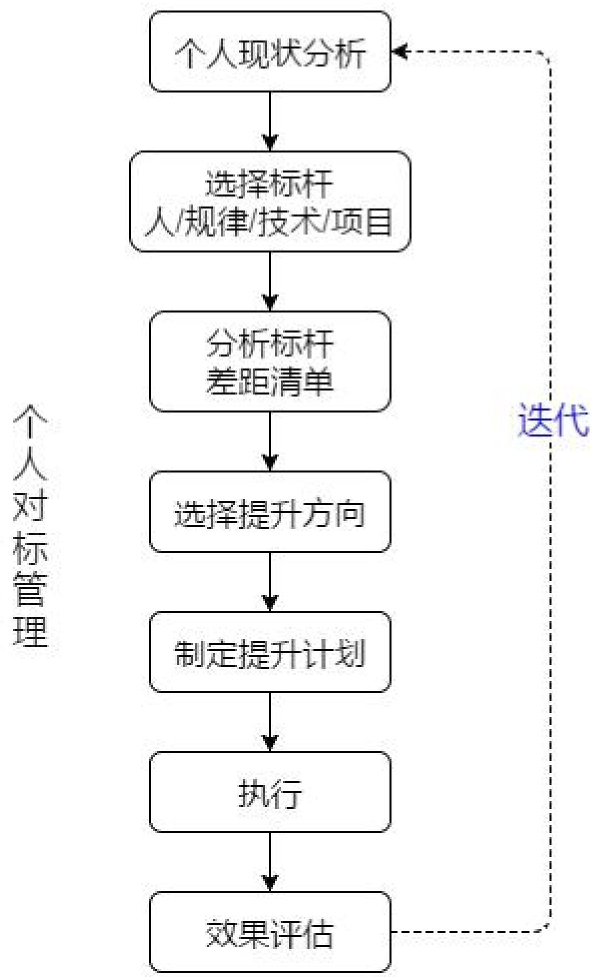


图2-3

以下解释来自百度百科：

对标管理，由美国施乐公司于1979年首创，均将其视为现代西方发达国家企业管理活动中支持企业不断改进和获得竞争优势的最重要的管理方式之一，西方管理学界将对标管理与企业再造、战略联盟一起并称为20世纪90年代三大管理方法。

对标管理是指企业以行业内或行业外的一流企业作为标杆，从各个方面与标杆企业进行比较、分析、判断，通过学习他人的先进经验来改善自身的不足，从而赶超标杆企业，不断追求优秀业绩的良性循环过程。

所谓“对标”就是对比标杆找差距。推行对标管理，就是要把企业的目光紧紧盯住业界最好水平，明确自身与业界最佳的差距，从而指明工作的总体方向。

在针对个人运用对标管理法时，可以从4个方面来寻找标杆：

- 优秀的人
- 一般性规律
- 技术本身的知识层次
- 项目指标

接下来我们就从这4个方面展开，看看怎么寻找我们的目标。

1. 从优秀者身上找目标

我们身边一定有人在某方面做得比自己好，比如：

- 张三的设计文档写得结构合理、条理清晰；
- 李四的UML图表画得准确；

- 王五对ES6标准掌握得好；

- 赵六擅长做代码管理；

- 钱七的架构设计能力超群，对产品的架构如数家珍；

- 毛八每天上班前都会列出要完成的三件事，每天下班时都会总结；

- 胡九学习新技术特别快，总是在项目组中担任技术预研角色。

……

别人做得好的方面，都可能是我们努力的方向。我们要用善于发现的眼睛，找到身边人的突出之处。

在向优秀者对标时，下面的问题清单可以帮助我们有序、系统地分析标杆：

- 他在什么事情上做得突出？是怎么做到的？

- 他有哪些知识、技能是我不具备的？

- 他有哪些提升效率的工具？

- 他有哪些好的工作习惯？

举个例子。

袁大每天都能准时下班，工作还完成得很好。你对这一点很感兴趣，就暗中观察他是如何做事的，发现他过一段时间就会翻看一下纸质笔记本，或者用笔在本子上记录点什么，还有，每天下班的时候，他都会在本子上写点东西。

于是你跟他聊天，发现他每天下班前都会在笔记本上记录今天完成了什么、遇到了什么问题、明天做什么。还了解到他每天都会早到半个小时左右，利用这段时间规划一天的工作。

后来你明白了，袁大培养了一个“早规划晚回顾”的工作习惯，通过这个习惯，保证每天都有几件重要的事可做，每天都有目标、有节奏，这样就可以不慌不忙地工作。

于是你就会思考：袁大的习惯是否可以成为我的习惯？

这个时候，你就找到了一个提升的方向：培养每日完成三件事的习惯。一旦你养成这个习惯，习惯的力量就会帮助你集腋成裘，完成从量变到质变的过程。

再举个例子。

你发现组里的袁二，排查Bug特别厉害，像一休哥一样，点点头沉思一下，就可以说出问题所在。即便是别人的代码引入的Bug，他也可以很快找到原因——只需要翻翻代码，和对方聊几句。

为什么袁二这么牛？

• 你向他请教，发现他做到了以下几点：

• 对业务特别熟悉，非常清楚某个业务到底是什么，用户在软件上怎么使用这个业务。

• 对业务逻辑和代码的映射关系特别熟。

• 爱看代码，所有人的代码他都看。

好了，正好你总是被Bug困扰，往往一个Bug能让你愁烦一个星期，这下是不是有努力方向了？

2.一般性规律

所谓一般性规律，指的是那些通用的、可以指导我们在什么时候做什么事情的规律。

举个例子，舒伯的生涯发展阶段理论就是一般性规律，男大当婚女大当嫁也是一般性规律。

对于开发者来讲，要关注专业能力成长的一般性规律，即技术成长三阶段。如图2-4所示。

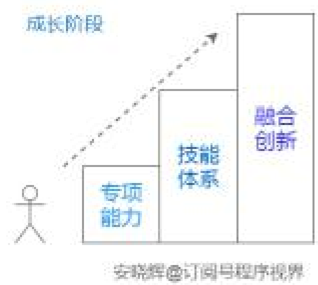


图2-4

在技术领域内的成长，基本上都会经历三个阶段：

• 第1阶段，专项能力的提升，这是初级阶段，你为了做事情，必须先具备某些基础能力，比如你要学会Python、Visual Studio、Vue.js、TensorFlow或者MyBatis等。

• 第2阶段，技能体系的构建，这是中级阶段，你拥有了一组技能，围绕某个方向构建了自己的知识图谱，能够用自己的方式来解决问题。比如在C++这个方向上，你用C++、Qt、OpenGL、Libevent、FFmpeg、WebRTC等组成了自己的知识图谱，可以胜任流媒体方面的产品开发。

• 第3阶段，融合创新，这是高手阶段，你具有了丰富的实践经验，具备了T型知识结构，形成了自己的思维框架和解决问题的框架，

能够融合不同领域的知识，组合各种资源，创造性地解决各种问题。此时你跳出了具体的技术束缚，站在了更高的层面，用底层认知和思维来指导你的工作。

对开发者来讲，若拥有一年左右经验，多数人处在第一个阶段——专项能力提升的阶段，熟悉某种编程语言，可以完成别人安排的一个小模块的开发。

若拥有三年及以上的经验，就应该进入第二个阶段了。当你在某个技术方向上构建了技能体系后，就可以完成相对复杂的工作，可以独立做一些事情，甚至可以辅导初级开发者来完成工作了。这个时候，你往往已经是团队里富有生产力的成员了。

若有五年以上开发经验，应该进入融合创新阶段，能够独当一面，可以独立完成特定项目的评估、设计、技术方案选择等事情。此时你往往是团队里的技术领袖或者技术管理者，具有比较大的影响力。

假如一个开发者干上八年十年，还到不了第3个阶段，可能就需要考虑通过其他方式来提升自己的竞争力，保住自己在团队中的位置。

这个模型更适合应用开发人员，对于做基础研究（比如音频算法、图像处理算法等）的开发者，在第3阶段，可能在他所在领域内钻得更深，成为专家。

我们了解了技术成长的三个阶段，就可以结合自己的工作情况，判断自己当下处于哪个阶段，该做什么事情了。

比如你做了2年PHP开发，可能你处于从第1阶段向第2阶段转型的过程中，此时提升的方向，就可以考虑和PHP相关的技术栈，比如了解HTTP服务器是如何和PHP整合在一起的，再比如了解数据库、操作系

统，这样你就可能会定下掌握LAMP（Linux/Apache/MySQL/PHP）或者LNMP（Linux/Nginx/MySQL/PHP）技术栈的目标。

3.技术本身的知识层次

一门编程语言、一个技术框架，其本身的知识层次也会有深浅，在学习时，也存在先后顺序和一般性规律。从这个角度讲，技术本身的深浅层次也可以用于个人对标管理。

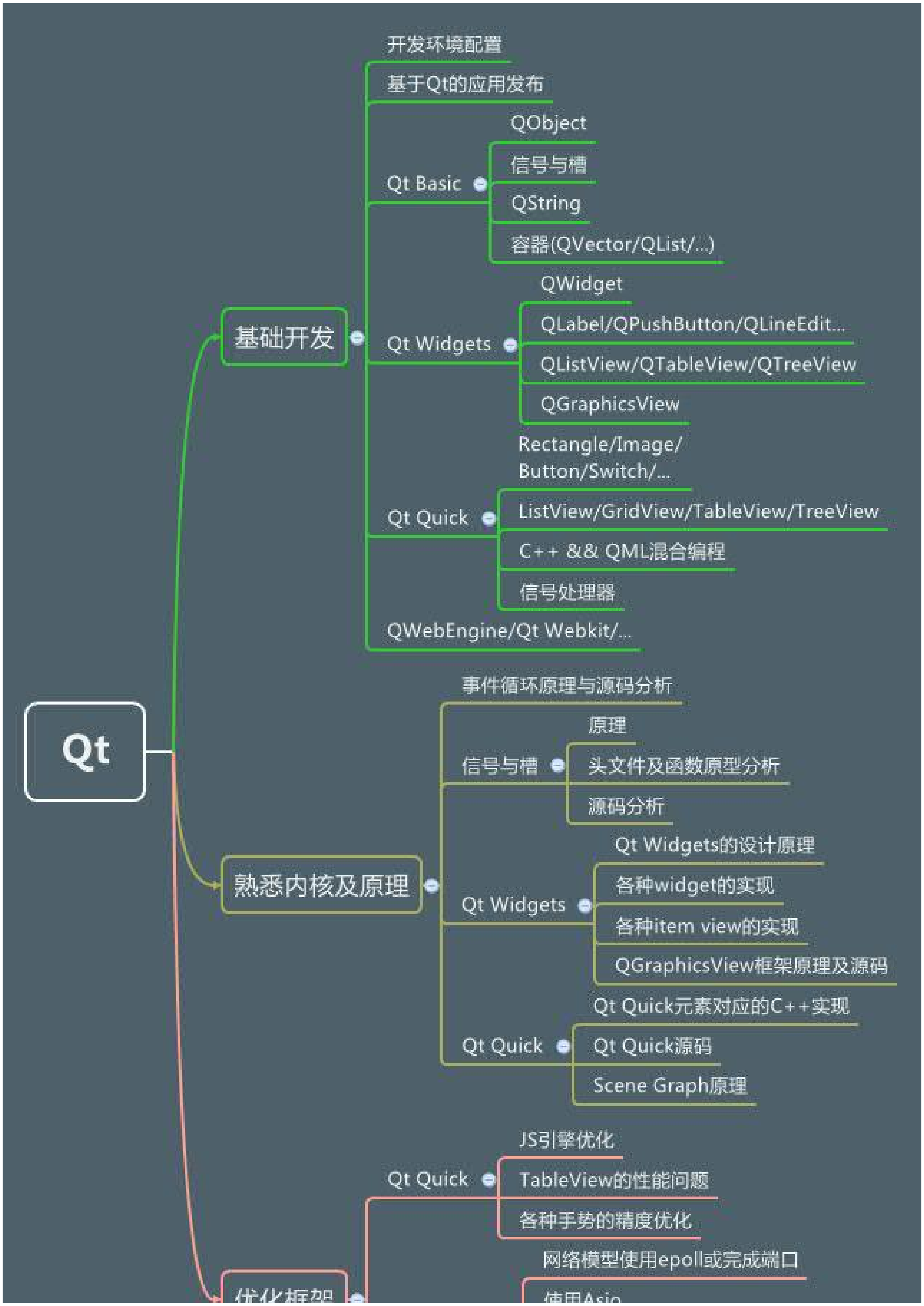
一般来讲，学习一门技术时，有三个阶段：

- 第1阶段，基础开发，了解API，基于API开发应用。

- 第2阶段，熟悉内核及原理，主要是了解框架的设计原理，阅读源码，洞悉内在机理。

- 第3阶段，优化框架，主要是针对框架已有功能的不足进行完善、优化，或者使用框架提供的机制扩展框架功能，或者对框架进行定制，让它适合特定情境。

我比较熟悉Qt，Qt这个应用开发框架，三阶段的划分可能是图2-5所示这样的。



Qt
基础开发
开发环境配置
基于Qt的应用发布
Qt Basic
QObject
信号与槽
QString
容器(QVector/QList/...)
Qt Widgets
QWidget
QLabel/QPushButton/QLineEdit...
QListView/QTableView/QTreeView
QGraphicsView
Qt Quick
Rectangle/Image/Button/Switch/...
ListView/GridView/TableView/TreeView
C++ && QML混合编程
信号处理器
QWebEngine/Qt Webkit/...
熟悉内核及原理
事件循环原理与源码分析
信号与槽
原理
头文件及函数原型分析
源码分析
Qt Widgets
Qt Widgets的设计原理
各种widget的实现
各种item view的实现
QGraphicsView框架原理及源码
Qt Quick
Qt Quick元素对应的C++实现
Qt Quick源码
Scene Graph原理
Qt Quick
JS引擎优化
TableView的性能问题
各种手势的精度优化
网络模型使用epoll或完成端口

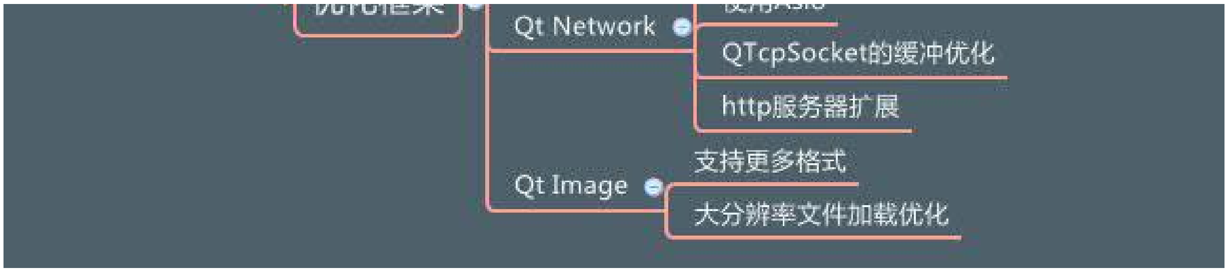


图2-5

多数技术框架，通过分析，都可以划分出类似上面的知识层次和学习阶段。

以此作为对标的标杆，就可以弄明白每个阶段应该达到什么程度，还可以定位自己处在哪个阶段，当前阶段的任务有没有完成，接下来该学什么。

4.项目指标

开发者的工作往往是由一个又一个项目串起来的，每个项目都会有预期结果，都会界定怎么样才算是完成，然后会有一系列的指标用于衡量项目做得怎么样，比如Bug率、延期时间、并发用户数、持续运行时间、单元测试覆盖率、安全性等。

我们在做项目时，就可以用这些指标来要求自己，这样每个项目都有目标，都可以制定一些策略，帮助自己来实现这些目标。

很多开发者其实不太关心交付时间、Bug率、冒烟测试通过率、并发用户支持、内存占用、CPU占用、电池消耗等问题，往往是做完了，能跑，觉得就可以了。

以这样的态度来应付开发任务，其实损失最大的是自己，因为你白白失去了锻炼和提升的大好机会。

如果以项目指标来要求自己，把项目指标分解到开发工作中，并且在开发过程中贯彻执行，收获一定比被动完成任务多得多。

举个简单的例子，如果你用Java开发一个电商类的Android App，内存占用就应当是你关心的一个指标，否则你的应用就会经常出现OOM错误，严重损害用户体验，导致用户大量卸载，最终影响产品的市场。

如果你把内存占用作为重点考虑的指标，一定会考虑如何使用图片预缩放、重用、解码格式、缓存等策略来优化内存占用，你甚至会自己设计一个图片缓存池或者特殊的ListView来专门处理用户快速浏览商品时巨大的内存消耗。

你有了这样的考虑，做出来的APP肯定比你从未考虑过内存占用问题而做出来的稳定得多。

拿来即用的自我提升方法

前面介绍了如何从4个方面寻找目标提升自己，只要你遵循那些策略，付出一些努力，就可以找到适合自己的提升策略。

下面给出一些经过验证切实有效的提升策略，你可以拿来直接用。

- **尝试用同一技术的不同模块或API来实现**，能让你更了解所用技术。

- **看看你正在用的技术，想想你处在三个层次的哪一层**，找到继续提升的空间，去学习、实践。持续这么做，能让你从泛泛的了解、基础的使用，进阶到熟悉、精通。

- **了解和当前所用技术相关的技术**，可以拓展你的知识图谱。

- **尝试用不同的技术来实现**，能加深对问题的理解，也能淬炼新的技术。

- **看看别人用的技术点、技术栈**，尝试了解，能拓宽你的视野。

- **看看同一项目内他人的设计和代码**，有助于理解整个项目。

- **尝试新的设计**，能加深对问题的理解，更能锻炼自己的架构和设计能力。

- **看看整个项目的需求、设计文档**。不要局限于自己负责的模块，这样可以提升全局观和系统观。

- **迭代式重构老代码**，迭代式重构可以解决时间不够用的问题。

- **阅读优秀源码**，看到好的，思考好在哪里，琢磨自己怎么做到，这样你就会日有寸进，终至千里。

- **参与开源项目**，参与开源项目比阅读开源代码的要求高得多，你要能够理解已有的代码，找到你可以做贡献的地方（issue、feature 等），你的代码要符合该项目的规范，还会被项目owner或其他成员Review，这些都是非常大的挑战，能让你快速成长。

- **写作技术博客**，有利于写作、逻辑思考、讲授、设计等能力的提升，也有利于系统化你的知识。最好的学习方式是输出。

- **讲给别人听**，锻炼讲授、演讲、沟通、归纳总结、逻辑思考等能力，对知识的内化与系统化也很有帮助。当你能够把一个知识点讲到别人也能听明白时，你就是真明白了。

- **与优秀的人和团队在一起**，镜像神经元会让你自动学习优秀者的做事方式，所谓“见贤而思齐焉”。

2.5 目标的设定与执行

当我们运用个人对标管理法从人、规律、技术、项目4个方面找到目标后，还要仔细地考虑两个问题：

- 这个目标适合自己吗？
- 如何完成这个目标？

1.适合性评估

先来看看如何判断某个目标是不是适合自己。有两方面：

- 这个目标和自己的职业规划是不是一致？
- 这个目标和自己当下的工作是不是可以关联起来？

设想你所在的团队里有位什么都可以搞定的全栈工程师，你非常羡慕这样的人，用对标管理法对他做了分析，发现他的知识图谱包括HTML、JavaScript、CSS、AngularJS、Node.js、MySQL、Redis、C等，那么，接下来，你要把他的技能树作为你的目标吗？

假如你也想成为一个全栈工程师，那么，你跟着他学习JavaScript前后端开发没有问题；假如你的目标是成为WebRTC领域的专家，那么，他的技术栈对你几乎没什么帮助，参考意义不大。

我们在运用个人对标管理法时，一定要理性地结合自己的长远目标，否则就会今天想学这个明天想学那个，久而久之什么也没学透。

所以，请先想想：**如果只能在一个技术方向上做到出类拔萃，你的选择是什么？**当你确定了这个方向，就拥有了主线剧情，可以运用

个人对标管理法来确保主线剧情充足发展，同时也可以在必要的时候引入支线剧情，辅助主线剧情的发展。

2.如何完成目标

当你选定了与你相关的某个目标后，如何完成呢？有两个关键点：

- 目标必须是有效的。
- 找到下一步行动。

1）有效目标

首先你要确保所选择的目标是有效的，符合SMART原则。

- S（Specific）：目标必须是具体的，要对标特定的工作指标，不能笼统。比如“我要学会前端开发”就不具体，而“我要学会HTML 5、Angular 4、Bootstrap 3，用它们做Web管理界面”就相对具体。
- M（Measurable）：目标必须是可衡量的，衡量的指标是数量化或者行为化的，验证这些指标的数据或者信息是可以获取的；比如“把Bug率控制在千分之三以内”就是可衡量的，而“软件没问题”就是非常模糊的说法。
- A（Attainable）：目标必须是可实现的，在付出努力的情况下可以实现；比如“在两周内学会HTML 5、Angular 4、Bootstrap 3”就是不太现实的。
- R（Relevant）：与其他目标有一定的相关性，比如你把代码规范化作为你的提升目标，就和你日常的开发工作有很强的关联性。

• T（Time-bound）：目标必须有明确的截止期限。必须有！没有期限，就没有目标！

以下是一个有效目标示例：

在三个月内学会HTML 5、Angular 4、Bootstrap 3，然后用一个月时间，采用SPA（Single Page Application）方式，开发清单APP的Web版本，支持登录登出、任务增删改、分组增删改功能，Bug率控制在千分之三以内。

2）下一步行动

仅仅拥有有效目标还不够，我们还要找到可以立即开始的下一步行动！

所谓“下一步行动”，就是某一件事情的下一个可以直接去做的步骤。

《小强升职记》一书中介绍了撰写下一步行动的4个秘诀：

• **动词开头**。一个好的行动应该是以动词开头的，比如“打电话给某某”“准备会议资料”“回复E-mail”等，以动词开头才能保证它具备可执行性。

• **内容清晰**。比如“准备会议资料”，虽然是动词开头，但是描述得不是很清晰，“需要准备哪些资料”“几点开会”“会议上要提出什么问题”，这些都需要进一步落实。所以说这样的下一步行动是失败的。

• **描述结果**。在任务开始之前对想要的结果进行描述，描述得越清晰，产生的能量就越大。比如这样：“早晨9点带着做好的计划书在1号会议室讨论营销计划，说服与会者认同我的营销方案。”

• **设定开始时间、周期、最后期限。**在设定了这三个和时间有关的属性之后，就可以更合理地安排自己的时间，把握行动的进度，照顾别人的时间。

如果你能够按照上述4个秘诀来拟定“下一步行动”，就有90%的可能性找到“可执行的下一步”。所谓“可执行的下一步”，往往是简单到你只需要迈出右脚就行了。假如你还要考虑到底是迈右脚还是迈左脚，就说明你的下一步存在未决因素，不能立刻开始。

下面是几个下一步行动：

• 找到AngularJS的官网。

• 买一本讲AngularJS开发的书。

• 买一本JavaScript的书。

• 2个小时完成Node.js下载与安装。

当你能从目标分解出能够立刻开始的下一步行动序列（最少三个）后，就可以做起来了。做完一个，分解新的下一步行动，加入到行动序列中，然后开始新的下一步行动。这样跑起来，你的目标就会稳步实现。

2.6 精进的4个习惯

习惯是很强大的力量，要把精进落实到日常习惯中。我个人有这4个习惯，供参考：

- 对标管理
- 三个问题
- 刻意练习
- 复盘

1.对标管理

前面仔细介绍了个人对标管理法的运用，它应该成为我们的习惯，成为习惯后，我们就可以自发地运用它，随时找到前进方向。

2.三个问题

参考《Scrum实战——敏捷软件项目管理与开发》一书，在Scrum开发模型中，有个每日站会。每日站会一般早上开，站着开（坐下来会让会议变长），用时10到15分钟。在每日站会上，每个人都要回答三个问题：

- 我昨天完成了什么。
- 我遇到了哪些问题。
- 我今天做什么。

回答完这三个问题，就可以更新看板上的任务状态，别人也都能了解到你的状态，如果有需要配合的，也都可以即时做出决定。

这三个问题，不仅可用于开发过程，还可演化成个人的工作习惯，指导我们每天的工作。

我是这么用的：

• 每天晚上下班时记录完成了什么、遇到了什么问题、明天准备做什么。记录在纸质的笔记本上。

• 每天早上正式开始工作前，审视昨天记录的内容，决定今天要做哪几件事（最好不要超过三件），今天就聚焦在这些事情上。

非常简单，但是好用。坚持这么做，你的工作就会越来越高效、轻松。最重要的是，你会知道自己每天都有成就，不会感到焦虑和恐慌。

3.刻意练习

推荐仔细阅读《刻意练习》一书。

我们可以把刻意练习简单地理解为4个要素：目标、Focus、Feedback、Fix it。

• 目标，每一次练习，都要有明确的目标，而且这个目标要高于你现在的能力，需要跳一跳才能够得着。个人对标管理法可以帮助你找到练习的目标。

• Focus，指专注地做事，所谓专注，就是方向明确，聚焦当下，心无旁骛，积极努力。

• Feedback，指反馈。我们在练习时，需要有高人陪伴，需要找到教练，能够及时给予我们准确的反馈，让我们知道差距。

• Fix it，指修正，改善。当我们获得反馈后，要根据反馈来改善自己的做法，这样才能进步。

下面是我绘制的刻意练习循环，如图2-6所示。



图2-6

一旦我们养成刻意练习的习惯，就可能在很多领域内成为高手。

4.复盘

参考陈中所著的《复盘：对过去的事情做思维演练》和成甲的《好好学习：个人知识管理精进指南》。

所谓复盘，就是在头脑中对过去所做的事情重新“过”一遍。它通过对过去的思维和行为进行回顾、反思和探究，实现能力的提升。

复盘分两种：

• 事件触发型复盘，比如项目抵达里程碑节点（或重大状态改变）。

• 周期性复盘，比如周、月度、年度、每五年。

下面这个清单，可以作为我们复盘的框架：

- 项目（事情）预期的目标是什么。
- 现状如何。
- 执行过程分析。
- 决定是如何做出的，有没有其他可能。

通过复盘，我们可以知道，事情结果比预期好还是坏，在执行过程中，有哪些环节做得好，哪些环节做得差，好的总结经验指导下次行动，差的反思原因制定提升策略，这样我们就可以获得成长。

很多开发者忙于做项目，往往是赶工、交付、开始新项目，很少去思考做过的项目做得怎么样，什么好什么坏，原因在哪里，怎么改进和提升，陷入马不停蹄做项目、一年经验用十年的怪圈，多年之后才发现自己的经验对不起工作的年限。

要告别这种状况，做项目时可以这样：

- 使用对标管理法为自己找到提升目标。
- 通过刻意练习来提升。
- 三个问题让你每天有目标，实现日有寸进。
- 运用复盘来成长。

2.7 习惯养成指南

可能有朋友会说，好习惯很难养成，想要一些好用的经验。

刚好，我自己琢磨过怎么养成习惯这件事。只要你能做到以下5点，习惯培养就会容易很多：

- 找到内在驱动力。
- 降低改变的难度。
- 让改变可视化。
- 奖励。
- 允许例外。

1.找到内在驱动力

有很多场景，会让你萌发改变的冲动。比如你去游泳，发现小伙伴有6块腹肌，而你肚子上是三圈轮胎，这时你可能就会说，我一定要健身，回去就办卡，每天都去健身房。但结果往往是，你在游泳时痛下决心，等离开游泳馆，回到家里，就觉得这事其实没那么重要，可以缓缓再说，然后又过了一天、两天，这事就更淡了，最终就不了了之了。

之所以会这样，是因为“养成健身习惯”这个想法，是被外在环境刺激而产生的，不是你发自内心的渴望，你想改变的动力是外在刺激，外在刺激消失后，你就失去了动力，就不会把改变落实下去。

同样，很多中学生每天坚持学习，堪称学霸，可一到大学，就丢掉了每天学习的习惯，浑浑噩噩过日子，成绩一落千丈，也是因为中学时天天学习并不是他自己渴望的，而是被父母逼的，所以到了大学，没了“父母逼迫”这个外在动力，他又没找到内在动力的话，就自然而然放弃学习了。

要想让一个习惯真正在自己身上落实，这个习惯必须是由你发自内心的渴望驱动的，只有你从自身出发，由衷地想要去做，你才可能养成这个习惯。

比如我们说的“三个问题”习惯，我说它可以让你日有寸进，积跬步至千里，你觉得挺好，应该有这个习惯。可是你并没有把它和你的某种由衷的渴望、欲望联系起来，那么你放下本书后，就把这个习惯也放下了。

但是如果你真的想要在一年内变得积极主动、工作高效，成为一个值得信赖的开发者，你就会琢磨：

- 这个习惯是否适合我？
- 早计划晚回顾，每天都有目标地去工作会让我变成什么样子？
- 我该怎么开始？
- 我该怎么持续，万一一周有一天我不想这么自律怎么办？

当你这样琢磨后，这个习惯就会和你产生联系，就会因为你想要变成专业人士而真的成为你的习惯，帮助你去实现自己的目标。

这就是你为养成这个习惯找到了内在驱动力。

2.降低改变的难度

很多时候我们无法养成一个习惯，源自于开始时给自己设置的难度太大。

比如你觉得晨型人都好有成就，都很出色，你就决定养成“每天5点起床，读一个小时书，跑一个小时步”的好习惯。可结果肯定会让你感到沮丧：太难了，根本不可能坚持下来，不行，做不到。

但实际上你分析一下就会发现，你所谓的“早起”，是三个习惯：早起、读书、跑步。你一下子想养成三个习惯，这就太难了。习惯最好一个一个来培养，养成一个之后，再开始另一个。

比如，你可以先养成早起的习惯：每天5点起床。起床后干什么先不约定，读书也好，做早餐也好，看美剧也好，都行，先让自己起得来。

当你连续21天（有种说法是21天养成一个习惯）都可以5点起床后，可以再来考虑起来之后是读一小时书，还是跑一小时步。

这个时候，因为早起已经不是问题了，没压力了，你面对的就只是坚持读书的压力。只面对一种压力，会比同时面对3种压力要容易得多。

有人觉得5点起床还是太难，那么，这个改变还可以降低难度。比如你现在早上7:30起床，那么你可以先比现在的起床时间提前20分钟，坚持一星期，再提前20分钟，每次提早都坚持一星期，直到起床时间变成5点。

其他的习惯也是类似的。以每天三个问题这个习惯为例，如果你一开始做不到每天早计划晚回顾，可以按周来做，每周一规划这周的事情，每周五回顾总结。这样难度就会降低很多。

等你习惯了每周计划、回顾后，再来调整频度，比如每两天、每一天。

3.让改变可视化

即便你找到了内在动力，想长期坚持做一件事时，还是会有很大的困难。你会觉得枯燥，会怀疑自己所做的事情是不是真的对自己将来的目标有帮助，甚至会因为长时间看不到结果而放弃。

要走出这种状况，提高成功率，一个切实可行的方法就是：让改变可见。

比如女生为了拥有好身材，决定养成“过午不食”的习惯，那么家里一定要放一个电子秤，每天称一称，这样才能看到体重减轻的效果；还可以买几条25（1尺9寸）的牛仔裤穿，腰部的松紧程度可以马上让自己感觉到身材的变化，引发过午不食的动力。

再比如说“三个问题”这个习惯，你想让改变可见，就可以：

- 用一个纸质笔记本来记录每天的计划和回顾的结果（纸质记录更容易让人有成就感，比电子的更容易看得见变化）。
- 专门用一个纸质笔记本记录自己每天完成的事情（这就是《小狗钱钱》中所说的成功日记）。
- 把两个笔记本放在工作台上可见的地方。

这样你每天上班都可以看到这两个笔记本，每次翻开笔记本都可以看到自己的记录，当你看到自己已经做了这么多天、看到每天都有完成的事情后，你就会觉得：哦，原来我已经做了这么多，原来我已经做成了不少事情。

这样的感觉会激励你继续前进。因为我们都渴望正向的激励、愉悦的感觉，你本能地会追逐这种感觉，况且，这种感觉还通向你的未来目标！

4.奖励

如果你打过游戏，一定对练级、过关、打BOSS有深刻印象。每当你打死一个小怪物、升一个级别后，都会有金币、装备、头衔等方面的奖励，让你看到，哇哦，我到68级了！

这就是奖励带来的愉悦、兴奋和刺激！你会渴望再受到这样的刺激，然后你会继续玩下去……直到防沉迷系统提示你。

养成习惯时，我们可以借鉴游戏中的奖励机制。

当你完成一件事情、坚持了一星期以后，就可以给自己一个奖励，比如一个笑脸贴（攒够10个可以换奖品），或者玩一局狼人杀，或者看一集《楚乔传》，或者给自己买支古风书签，给自己买个机械键盘。

这样你的坚持就会更好玩儿，更有趣，更值得期待。

5.允许例外

我建了一个早起打卡群，小伙伴们每天都在群里打卡，迟到或漏打一次要发50元的红包。不发则退群。

每位朋友加入时，我都告诉他规则，在他看了规则并同意之后，我才拉他入群。

可还是有不少朋友，迟到后选择退群，放弃坚持。

大部分这样退群的朋友，其实是对这个早起的习惯无所谓，并不真正想要。

也有一部分朋友，不允许自己例外，觉得自己迟到一次，美好形象就毁了，就没脸见人了，于是干脆放弃好了。

对于后者，我想说的是，你在养成一个习惯的过程中，一定要允许自己有例外。

比如早起习惯，偶尔睡过头也没什么关系，明天接着早起就是了。只要你在一个较长的时间周期内，大部分时间都坚持早起就可以了。一个月有一两次没按时起来，没什么关系。

再比如我们的“三个问题”习惯，你周六、周日就想娱乐，不想那么有计划性，没关系啊，尽情放松就行了。

我们养成一个习惯，目的并不是追求这个习惯本身100%准确，而是利用习惯的力量，让我们的工作像自动化脚本一样高效执行。

偶尔有一两次例外没什么关系，修复一下，继续执行，只要这些低频的例外不影响整体的效果，就不用纠结。

2.8 超越技术层面的核心竞争力

我们在第1章介绍过技术成长三阶段模型，它描述的第三个阶段，是融合创新阶段。在这个阶段，你建立起了可迁移的核心竞争力，使得自己拥有了超越技术层面的、快速解决问题的能力，即便到一个陌生的领域、原来的技术积累不能直接应用，也可以应用自己的思维、想象力、架构设计、分析、解决问题的框架等，快速厘清问题，找到解决问题的关键。此时你已经能够从容应对日新月异的技术变迁和各行各业加速融合的现状，因为你有了驾驭技术的“内功”。

这个“内功”，就是《你要如何衡量你的人生》一书所说的“应用流程”能力。

人的能力分为三类，如图2-7所示。

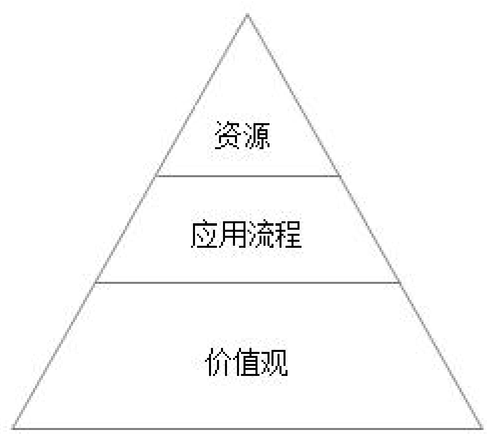


图2-7

所谓资源，包括知识、技能、时间、金钱、人脉、天赋等。

对于程序员来说，Java、C++、PHP、Spark、Scala、Qt、Node.js、Hadoop、Vue.js等，都是资源层面的能力。

资源往往是显性的、外在的，非常容易被自己和别人感知。

我们经常在招聘信息中看到的任职要求，比如精通J2EE、精通SSM（Spring+SpringMVC+MyBatis）、精通FFmpeg、精通Qt等，都是对资源层面能力的要求。

所谓应用流程，指的是解决问题的方法、思维的框架、分配资源的方式、自我管理的模式等。

应用流程是内在的，容易被人忽略，但它却是解决问题的真正能力，当你拥有应用流程能力之后，才能很好地利用资源来解决问题，创造价值。

放在开发者身上，自主学习、逻辑思维、数据分析、价值判断、优先级排序、框架设计、想象力等能力，就是应用流程方面的能力。应用流程这类能力，是可迁移的能力，是核心竞争力。有了这些，你才能快速地学习新知识新技能，才能更好地整合自己的知识、技能来解决实际的问题。

我们经常会看到有些资深开发者或者研发经理，只懂一种语言和相关框架（比如C++），却能够帮助Java开发者定位问题，甚至可以撸袖子上阵帮忙搞Java代码，就是因为他们已经超越了具体技术的执行层面，上升到了思维层面，拥有了应用流程方面的能力，并且能够应用这些能力，先在高纬度定位问题，然后再降维进入技术层面去解决问题。

当我们拥有了应用流程能力后，就会有举一反三、触类旁通的表现，落实到技术上，就可以一门精、多门通，形成T型的知识结构，拥有强大的解决问题能力。

所谓价值观指的是，你觉得什么是重要的，你要什么、不要什么、如何做决策。

一个人的行为价值观，是其最根本的能力。价值观决定了你在哪个方向做、以什么原则去做。没有价值观，你就会像浮萍一样，东飘西荡，很难做出有长远意义的选择，就很难形成有效的应用流程，也很难积累能被社会感知的资源。

热衷于技术的程序员，觉得技术成就更重要、自己解决问题更重要，他希望自己在某个方面具有顶端优势，他会一直做，不断复盘，不断精进，直到成为专家。

希望做管理者的程序员，认为领导别人很重要，职场权力和位置很重要，他往往在做上三五年后就会转移重点，放弃在技术上的深入积累，转而寻求管理职位的机会。

这两种不同的选择就是由不同的价值观决定的。所以，在决定将来的方向时，一定要先想想：

- 想要什么。
- 哪些人事物对自己是最重要的。
- 想成为什么样的人。

这是根本性的问题，答案指明你的职业发展方向。

当你的价值观指向技术、你的应用流程的能力淬炼到家后，你就具备了真正的核心竞争力，到这个时候，如果你在一个技术领域内有了深入积累，再接触新的语言、框架、技术，在应用流程的加持之下，很容易就可以一通百通。

所以，当你学习使用某一技术时，着眼点应该放在如何运用它来解决问题，如何在解决问题中淬炼自己的应用流程。这才是打造核心竞争力的正确姿势。

搞明白了开发者的能力模型，我们就可以来讨论一个很多人都关心的问题：**软件开发到底是不是吃青春饭的。**

所谓青春饭，着眼点在青春。你年轻，有体力，有时间，能拼，能加班，你依赖时间和体力的复制来进行工作，当你年龄增长时，时间和体力的复制不可继续，这样的工作，就是吃青春饭的。

但软件开发不是。软件开发有两个层次：**体力化的代码打写（编码）与思考层面的创造**。我把程序员的工作过程绘制成图，可以简单明了地说明这一点，如图2-8所示。

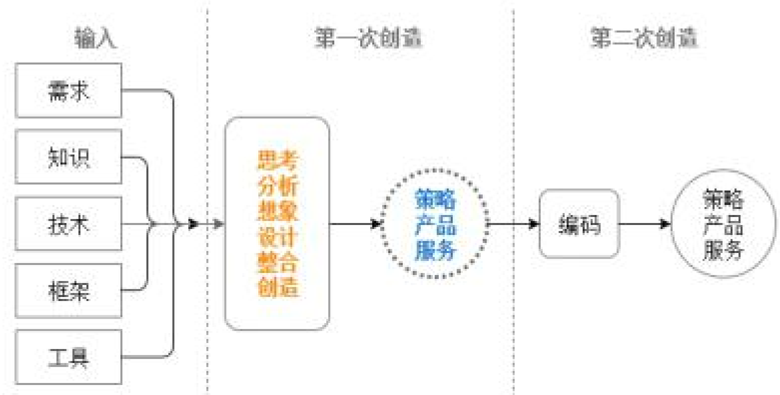


图2-8

抱持“软件开发是吃青春饭”这一观念的人，只看到了软件开发这一工作中“体力化的代码打写（编码）”这一初级层面，误以为程序员只能靠拼时间和体力来建立竞争优势，但实际上，体力化的优势，在偏重信息处理的工作中，永远都不是解决问题的关键，都不能带来竞争优势。

决定程序员是否具有优势的，是应用流程方面的能力，即逻辑能力、思维能力、想象力、架构设计、解决问题的框架、快速学习、自我管理、创新与创造等。

这些能力，都需要时间的沉淀和项目的历练，这也是我们的技术能力三阶段模型最后一个阶段应该拥有的能力。你只有经历了初级阶段、中级阶段，走过了5~8年甚至更久的时间，才能在实践和反思中培养出这些能力。而到这时，你往往已经是大龄程序员了。

不过不用担心，虽然年龄渐长，但你的优势也建立起来了：你的综合性想象力、逻辑思维、数据分析、解决问题的框架等能力都得到了充分的历练和实践的检验，遇到问题时能够更快地找到更合适的解决方案，先人一步在头脑里创造出最终的形象（第一次创造），然后在充分思考后用代码实现出来（第二次创造）。

由此看来，大龄程序员应该在实践中发现并淬炼自己的“应用流程”，建立相应的优势，这样，他就可以超越体力层面的局限，让自己的能力与价值随着年龄和阅历的增长而增长，就可以多多通过思考层面的创造来进行工作：**谋定而后动，做得更少，但更关键，也更好。**这才是更高效、更经济、更有价值的工作方式。

最后，有两点非常重要，提醒一下：

- 能做到这一步的大龄程序员其实是少数人。

- 有很多公司，用人急功近利，看不到走到这一步的大龄程序员的优势。

所以，如果你成了大龄程序员，虽然你可以超越年龄的限制，还是要尽量找到懂你价值的公司、懂你价值的团队，只有在这样的重视技术价值的环境里，你的价值才能得到体现。

2.9 公司内的职业规划

本章一直在和大家探讨如何使用个人对标管理法在技术上持续精进，但实际上这只是开发者职业规划的一部分内容，开发者的职业规划，还包括职场目标的设定、晋升通路的设计等。所以，下面我们就来提供一个简单的职业规划模型，你可以在公司范围内应用它来指导你在组织内的发展。

图2-9对这种简单的职业规划策略做了形象化展示。

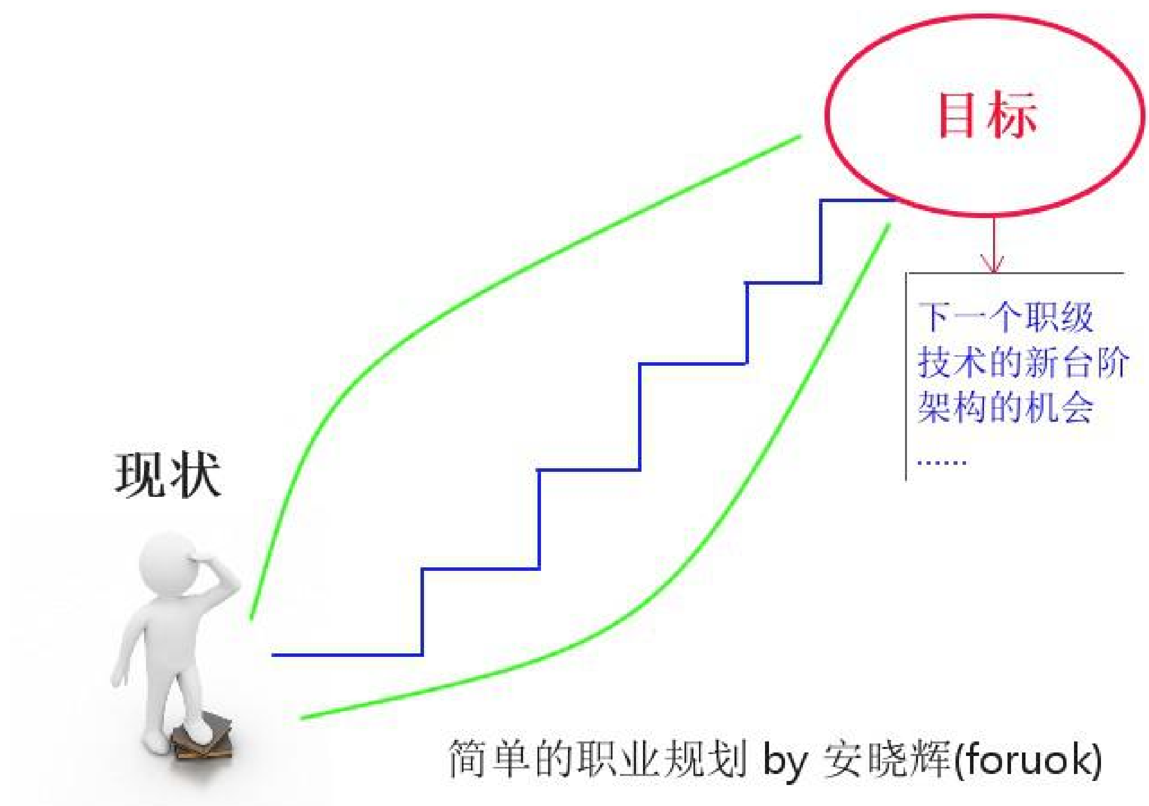


图2–9

根据这个简单的职业规划模型，你只需要弄明白两个问题，就可以导出目标和工作计划，然后就可以通过执行计划，达成自己的目标。这两个问题是：

- 自己的下一个台阶在哪里？技术、架构、职位

- 怎样做才能站上下一个台阶？精进技术、文档、设计、组织……

那么，怎么才能找到你的下一个台阶呢？前面介绍过的个人对标管理法可以帮助你寻找技术或架构上的精进目标，职位目标可以从公司内部的职业发展通道设计中找到。

很多公司都对各个职级有描述，比如阿里的工程师，职级从P3~P10，管理人员，职级从M1~M5每一级的要求都有，可以查到。

比如你是P5（高级工程师），那么你的下一个台阶就是P6（资深工程师，相当于M1）。去研究这一级别的岗位定义和职能描述，看看它需要什么知识、技能、经验，有针对性地去准备。

这就是企业内的职业规划策略。只要你始终向前看，就会不断成长。

时间上不用看太久，当下的半年、一年就好。因为人的想法时刻在变，应该根据外在环境的变化带来的新机会不断评估自己的方向。

第3章 成为技术管理者

很多程序员做了2~3年开发工作后就会考虑这个问题：要不要转管理岗位？

你考虑过这个问题吗？

当你开始考虑这个问题时，会遇到更多的问题：

- 哪些职位属于技术管理？
- 做管理一定比做开发有钱途吗？
- 如果做管理让自己失去技术竞争力该怎么办？
- 自己是否适合做一个技术管理者？
- 成为一个技术管理者都需要哪些能力？
- 管理岗位都做哪些事情？
- 自己该怎样为成为技术管理者做准备？
- 走向技术管理有哪些方式？

这些问题都是本章要讨论的话题，你将从中找到自己想要的答案。

本章会先探索一下怎么判断自己是否适合从事技术管理岗位；然后尝试着理解管理角色，理解技术管理人员都要做什么事情，需要什

么能力；最后会讨论你最关心的问题——走向技术管理的常见方式，以及怎样为成为管理者做准备。

3.1 真的要做管理吗

在很多人眼里：

- 不想做管理的程序员不是好程序员。

- 不做管理的程序员没前途。

这也是我放弃研发经理的Offer，选择回归开发岗位时面临的环境：不少朋友从上面两点出发，认为我做了愚蠢的选择。

但实际上，正如我们在上一章——如何在技术上持续精进——的一开始所介绍的方法——成就感来源中所说：

每个人的成就感来源都不一样，假如你像我一样，最关注的点在于自己动手解决具体的技术问题，那么开发工作就更适合你；假如“领导和管理别人、通过别人完成工作、看到别人成长”让你更有成就感，那么管理工作适合你。

2016年7月12日，有位朋友在微信上和我聊天。

ZX：晓辉大哥好，最近很苦恼啊，干了10年安全相关的开发工作，现在被转到管理岗位了，带着10多号人，比较头大，想转回去做开发了。

我：可以啊，做开发挺好。我现在也不做管理了，做开发。

我：你肯定是因为技术做得好，所以被转管理喽。

我：领导可能以为这样的晋升是对你的激励。你可以找领导谈谈，沟通一下你的倾向。

ZX：其实以前也聊过，说自己不太适合带团队，领导说我没问题。只是我现在干得比较不爽，找不到成就感，活派下去了，我闲了，不知该干啥了。

ZX：而且这种带人的事情干得比较别扭，我不习惯安排别人做事。

这位朋友因为技术做得好而被晋升到了管理岗位，但他很明确他喜欢做技术，喜欢专业带来的成就感，所以他在想着如何逃离安排给他的管理工作。

2014年6月13日，某知名视频网站研发总监老陆在接受CSDN采访时，回答了“为什么转型做管理”这个问题，他说：

其实，做程序员挺好的，学会一种新技术或者解决一个技术难题，很容易就能获得成就感。但做了3到5年之后，你就会觉得迷茫：似乎该会的也都会了，再多学点别的吧工作中又用不到，我的前路在何方？现代的软件都不是单靠一个人能做出来的，必须靠团队。个人做得再好，也未必能做出更好的产品。我当时就意识到，个人技术再牛又怎么样呢，让团队发挥最大的效力一起把事情做好则是另外一门学问。那才是我要走的路！

我从带4到5人的小团队开始，从负责底层的SDK开始做起，后来负责整体的应用软件，负责从开发、测试到产品发布的整个流程，团队规模也越做越大。工作越来越忙，管的事情越来越杂，慢慢地不再有时间写代码。在这个过程中，我得到了很多人的帮助，自己的能力也得到了长足的发展。自然而然地，我不再是一名程序员了！我已是一位职业的管理者。

老陆对技术管理工作有清晰的认识，从事技术管理工作，能够带领团队完成更大的目标，做出更好的产品，它需要你慢慢放弃对技术执行层面成就感的追求，转而把自己当作催化剂，来激发别人的潜能，让团队发挥最大的效力。

这需要对职能定位有明确的认识，因此，我们接下来看看常见的5大职能定位，对它们的描述，可以帮助我们找到自己的方向。

3.2 5大职能定位

常见的5大职能定位：

- 专业技术者
- 自由职业者
- 管理者
- 创业者
- 投资者

1.专业技术者

专业技术者通过将自己嵌入组织中，运用自己的专业技能解决问题，其价值通过组织的产品或服务体现。他们需要接受组织的规章制度约束，但是能够获得稳定的工作环境和保障。

专业技术者享受亲自执行、亲力亲为地解决具体问题，这样他们会感到自己有价值、有存在感，进而会有成就感。

专业技术者通常为成为一家公司的雇员、员工，每个月固定时间拿薪水会让他们有安全感、稳定感，觉得生活有保障，薪水之外的福利，诸如月饼、超市购物卡、粽子、米面油、年会抽奖、项目奖金、年终奖等，会让他们津津乐道。

不确定性会让他们感到不快乐或者恐惧，稳定性、确定性和一致性才能让他们安心。

大部分开发者都是这类专业技术者，希望在某个组织内工作，拥有好的薪酬、福利，获得好的职业发展。

2.自由职业者

自由职业者喜欢为自己做事情，希望自我支配，不想受组织约束，渴望自由和独立性。

他们喜欢努力就有回报，崇尚多劳多得。另一方面，他们不喜欢由别人来决定他们能挣多少钱，更不喜欢由不如自己有能力又不如自己努力的人来支配他们。

所以自由职业者会脱离组织，自己当自己的老板。他们直接面对客户，向客户提供自己的产品和服务，自己决定以什么方式赚钱、赚多少钱，自己决定何时工作、何时休息。

但是自由职业者的收入是否稳定，依赖于自己对客户的开发与维系，很可能充满波动性，这个月分文皆无，下个月盆满钵满，这都是常有的事。

自由职业需要极强的自我管理能力。

自由职业最大的优点是自由、做自己想做的事、按自己想要的方式做事。但反过来看，为了能够赚到足够的钱，他们也需要看客户的脸色来行事，并不享有绝对的自由。

自由职业者首先是专业技术者，对自己在专业方向上的能力、成就和声誉非常在意，他像专业技术者一样，享受亲自执行、亲力亲为解决具体问题带来的成就感和价值感。

我个人曾经做过十几年开发工作，现在成了自由职业者，在我辞职成为自由职业者时，有时一个月都没什么收入，最初一年，平均每

个月的收入，不足我做开发时的1/4。但是我更看重自由、独立性、做自己喜欢的事及用自己的方式做事，愿意为此承担一些代价。这也是很多人选择自由职业的理由。

3.管理者

和专业技术者一样，管理者也是组织的雇员。不同的是，专业技术者在执行层面做事，亲力亲为，专注于如何做好执行层面的任务，而管理者通过他人完成工作，把组织目标拆解成执行层面的任务，委派给专业技术者等执行人员，通过组织、管理、计划、激励、反馈等管理策略，领导他人完成任务，实现组织目标。

管理者着眼于通过领导、组织、协调、管理等手段来驱动他人达成目标，他们更看重的是如何通过他人完成工作，而不是自己亲自做执行层面的事。

开发团队中的管理者，尤其是一线的技术经理、研发经理，往往拥有多种身份，既要懂技术，又要会管理，和宽泛意义上的管理者略有不同。

我2009年开始做技术管理，后来做研发部门经理，直到2014年。在我做管理的时候，一方面要做项目管理、人员管理，另一方面还会花至少1/3的时间来做技术。我会给自己一些不在关键路径上的开发任务，会做一些新技术探索，会做些架构设计的事情，这样可以保持对技术的敏感，能够始终了解一线开发者的工作和想法，可以更好地帮助我完成管理工作。

4.创业者

创业者特别想要拥有自己的产品或服务，特别想建立自己的企业，特别想通过自己的产品、服务、企业实现自己的价值，建立自己的影响力，获得成就。

创业者往往会成为企业所有人，如果做得好，还会成为出色的企业家。

创业者必须具备远见、定见、勇气和韧性，在成为企业家的路上，他们还需要不断锤炼自己的领导艺术和商业技能。

5.投资者

投资者自己不拥有企业，他们通过投资别人的企业来获取收益。他们相信资本的力量，让钱为他们工作。

很多创业者成功后，会转身成为投资者，兼具企业家和投资者双重身份。比如360的周鸿祎，新东方的俞敏洪，当当网的李国庆。

你可以仔细研读上面5种职能定位的描述，看看自己更倾向于哪一种；然后回顾自己过去的工作经历，找到让你有成就感的事情，看你的成就感来源，是亲自做事还是领导他人做事；这样就可以大致判断出来你是否倾向于做管理工作。

需要特别注意的是，从开发者到管理者，并不是简单直接的职位晋升，而是一种转型！后面我们在第8章可以看到，这种晋升，发生了职能转换。开发者更多的是做事，亲自解决具体问题，而管理者，更多的是领导团队做事，通过他人完成工作，以团队的成绩来界定自己的工作结果。

但很多技术人员（执行者）在初任管理职位时，往往转不过这个弯儿来。他们会继续紧抓技术，企图用自己的技术优势来奠定自己的领导力，凡事都要亲力亲为，反倒是对领导、组织、激励这些与人相关的事情能躲则躲、能拖就拖，往往到了不得不做时才被迫去弄一下。

这样的结果呢，往往是团队没带好，领导对你也不满意，而你觉得自己这么认真做事，付出这么大的努力，却没好结果，心里感到委屈。

其实这是角色意识转换的问题。作为管理者，要把更多的精力放在“人”身上，通过他人来完成事情。你在人身上花费一分精力取得的成效，相当于在事情上花费四分精力取得的成效。

从执行到管理的转型，对很多人来讲，都会经历一段痛苦的时期。甚至你原来执行层面做得越好，痛苦就越多、越久——因为你要放弃依赖以往辛苦积累起来并被证明行之有效的优势。在下一章——技术管理新人面临的挑战中，我们还会聊到这一点。

3.3 理解管理角色

让我们再来理解一下管理角色。

管理者通过他人完成工作。这是管理者的定义。

管理者有两大任务：

- 完成工作目标
- 培养下属

有些开发者觉得自己性格内向、不会说话、不善于委派工作、还不好意思压别人干活，担心自己做不好技术管理工作，当了管理者后无法完成工作目标；同时还觉得自己不知道怎么辅导、培养别人，又担心别人不服自己，不和自己合作。于是就想做又怕做不好，非常纠结。

但实际上，管理者不一定是外向者，内向者也可以成为很好的管理者。肯尼迪、奥巴马、丘吉尔，都是内向者，但他们都是很棒的领导者和管理者。据统计，约40%的领导者实际上属于内向型性格。在软件领域，更是有大批内向的管理者，比例甚至超过40%。

老陆（《DirectShow开发指南》和《DirectShow实务精选》作者）是某知名视频网站研发总监，负责PC端产品研发团队，管理做得相当出色。

我们初次见面，他和我一样有些拘谨。聊天时，明显可以感觉到他话不多，三思而后言，偶尔沉默，情绪内敛，不大外放。通过这些

表现可以看出他也是内向的人。我在写作本书时，和他确认了这一点。

我也是一个内向者，更愿意把注意力和能量放在内部世界，偏好精神活动，喜欢先思考再行动，相对来讲比较沉默，看起来没那么热情。我偏好小范围的活动，害怕成为被注意的中心，是慢热型的，到一个新环境，或者经历一个新职位，总是需要比较长的时间来适应。但是我在2009年—2015年，一直做研发管理工作，多时管理40多人，也可以胜任。

由此看来，即便你有点内倾，依然可以尝试去做管理者。因为管理者不是天生的，是可以培养的，管理和沟通，都是技能，技能都是可以习得的。只要你掌握一定的方法，通过练习，一定能胜任管理岗位的工作。

如果你想做管理者，尽可以去尝试。但有一点尤其需要注意：管理者并没有看起来那么轻松。

1.管理者没那么轻松

很多开发者羡慕管理者轻松、钱多、职位高，因而想做管理者。但实际上，一个管理者，要想把工作做好，并没有那么轻松，甚至非常困难。

从表现上看，优秀管理者应当做到这几点：

• 下属可以各尽所能，都能得到成长，个人目标可以在实现团队目标的过程中得以实现。

• 下属信任你，愿意跟着你干。

• 实现团队目标。

· 上司信赖你，愿意把团队交给你管，愿意把复杂的事情交给你的团队去做。

这4点中没有一点是轻轻松松就可以做到的。你看到某个管理者轻松，他要么没做到这几点，要么是把自己的努力、用心、辛苦包裹了起来，故意表现出举重若轻的样子。

以第1点为例，你要想让下属各尽所能并有所成长，就需要认真、细致、用心地做4方面的事情：

· 了解每一个人的性格、能力、知识、技能、优点、缺点，还有他们为什么在这里工作，在这里求什么。

· 拆解团队的目标，形成粒度较小的任务。

· 结合团队成员的个人能力和目标，把任务匹配到个人，形成个人的绩效目标管理表（MBO）或者OKR表。

· 及时给予下属反馈。

我们只看“了解下属”这一点。

你要了解一位下属，就需要寻找各种机会，经常性地和他沟通，比如吃饭、闲聊、讨论技术等，还要定期一对一深度沟通，让他觉得自己是受重视的、特别的，这样他才可能信任你，打开心扉，给你看更多的东西，你才能了解他。

这是非常困难的一项工作，大部分技术管理者在这方面做得都不及格——他们只关注程序员有没有完成任务，而不关注程序员这个人，往往把程序员当作资源来看待。比如“哎呀资源不够，再招两个程序员吧”，诸如这种说法很常见。

《代码之道》中说："成为一名优秀的管理者，所有你要做的就是确保你的人能够工作，并且把他们当人（而不是资源）去对待。"

这话之所以被强调，其实是因为：关注人，真正做到关注某个人、关心某个人，真的是非常艰难的事情。

但反过来讲，如果你不真正关心一个人，就很难了解他，很难委派合适的任务给他，很难让他在完成工作任务的同时实现其个人目标，长此以往，这个人一定会感觉糟糕——因为每个人都希望自己是特别的，都希望得到领导的认可和赞赏，都希望能够在一个地方获得成长，而他从你这里得不到！

那么接下来，这位感觉糟糕的人，有机会就会离开。

如果这样持续下去，你的团队就没有积极性，产出率就会降低，就很难完成公司目标，陷入恶性循环。

如果看了我说的各种困难，你还是有志于管理好一个团队，认为自己可以创造一个能让大家各尽所能的环境，在实现团队目标的同时，也能让每个人都得到成长，那么你就可以考虑向管理岗位进军了。

2.常见的技术管理岗位

常见的技术管理者类型如下：

- 技术主管（经理）
- 项目经理
- 研发部门经理
- 研发总监

- 研发副总裁
- CTO

我们讨论技术经理、项目经理和研发部门经理这一层，更高级别的暂不讨论。

大公司，技术主管、项目经理往往是分开的，一个偏重技术层面的管理，一个偏重项目本身的管理。小公司往往是合一的，既要管技术，又要管项目。技术主管和项目经理，往往是随项目而生的，具有可变性，不是固定的职位序列。可能这个项目张三是项目经理，王二是技术经理，下个项目李四是项目经理，赵六是技术经理。

研发部门经理往往是一个研发部门行政意义上的职位，负责这个部门整体的技术、项目和人事管理。在有些公司，研发部门经理会在开发项目时，参与项目管理，此时则可能身兼技术主管、项目经理、人员经理三个角色。

技术主管、项目经理，往往是通向研发部门经理的台阶。

3.技术管理的职责

我们先了解技术管理者都要做些什么，由此就可以看到他需要什么能力。

怎么知道技术主管、项目经理、研发部门经理要做什么呢？有三种途径：

- 公司内的职级说明
- 招聘信息的岗位职责和工作内容说明
- 向技术管理者咨询（即生涯人物访谈）

技术主管、项目经理和研发部门经理的职责可以分为三类。

1）技术管理职责

- 技术方案评估与选择
- 关键技术决策
- 工作量评估
- 任务分解
- 委派任务
- 代码规范管理
- 代码审核
- 技术风险识别与控制
- 团队技术能力管理
- 关键代码实现
- 技术督导
- 技术培训
- 售前或售后技术支持

2）项目管理职责

- 项目中的人员管理与调配
- 项目计划制定

• 研发任务管理

• 项目进度管理

• 协调沟通

• 教练指导

• 复盘总结

• 组织间接口协作（测试、产品、需求、市场、销售、售前、售后、客户）

3）人事管理职责

• 招聘、面试

• 解聘

• 人员调配（包括调度、任免、角色安排）

• 资源协调

• 绩效考评

• 职级评定

• 薪水调整

• 管理制度评估

• 人员预算

• 财务预算

技术主管的职责偏重第1）部分，项目经理的职责偏重第2）部分，但有时也会有交叉。比如技术主管（经理）有时可能会兼管项目进度和计划跟踪；而有些团队只有项目经理而无技术主管，此时项目经理也需要负责方案选型、技术决策、工作量评估等工作。在很多规模较小的公司，技术主管和项目经理，往往是合一的。

注意，研发部门经理其实对其所辖部门的每个项目都负有责任，所以他往往是上面几种职责汇聚一身。

4.技术管理需要什么能力

技术管理，着眼点还是在“管理”二字，因此除技术方面的积累和见识外，更重要的，还是要习得管理者所需的各项能力：

- 共情（同理心）
- 委派任务
- 沟通
- 反馈
- 激励
- 目标统合
- 向上管理
- 时间管理
- 绩效评估
- 知人善用

- 规划
- 计划
- 组织
- 协调
- 管理
- 选择
- 责任
- 辅导
- 讲授
- 演讲
- 复盘
- 承压
- 勇于挑战

我们挑几种重要但作为执行者的程序员又不太具备的能力来展开描述一下。

1）共情

领导者不能把人当作完成任务的机器，一定要尊重下属，把下属作为活生生的、独特的个体来看待。作为一个鲜活的个体，他有倏忽来去的情绪，有自己的想法和看法，有自己喜欢与讨厌的东西，有特

定的家庭环境，所有这些，会让他和你不一样，和你预期的不一样。他一定不是你期望的那个样子，总会有这方面或者那方面和你的想法不一致，这些不一致，一定会体现到工作中来。

每一个下属都是独一无二的鲜活个体，都有独一无二的个人情况，作为领导者，一定要体谅下属，能够理解下属的处境，这样才能和下属建立相互信任的关系，工作才能在情理通畅的背景下进行。

而要体谅下属，领导就要具备共情的能力。

所谓共情，是指一种能进入到对方内心世界并体验对方内心感受的能力，然后将对对方感受的理解用自己的言语表达出来，使对方感受到被理解、被接纳。共情是从对方的角度，而不是从自身的参照体系出发去理解对方的能力。

简单说，共情就是换双鞋子走路，你要时常穿着下属的鞋子走走看，这样你就能体会到他的想法、问题和难处，就能理解他为什么最近工作消极效率低下，为什么三不五时要请假，为什么早上老迟到，为什么不按照你说的方法去做事……

假如你忽略下属的独特性，不能做到共情，就只能看到办事不力、态度不好、结果不符合预期、没有责任感等问题，而看不到背后的原因，就不能有效地激励下属改善自己的工作方法，取得更高的生产效率。

同样，我们也要把上司当成具有七情六欲的活生生的人，他有他的情感偏好，有他的习惯，有他的好恶，有他的职权和强势，也有他的难处和纠结，只有你把他当作一个带感情的个体，才能更好地理解他，摸透他的习性和工作风格，更好地与其协作。

职场上很少存在非黑即白的情景，也很少有人能只凭逻辑正确和就事论事就把工作做好，相反，每个人都是情绪化的，往往只有情感才能让一个人自发自愿做事情。而要撬动情感的力量，共情，就是一种必需的基础技能，有它的加持，工作才能越做越顺。

2）委派任务

假如你现在是技术经理或项目经理：

• 你能明确区分出来哪些事情是你必须做的、哪些事情是其他团队成员必须做的吗？

• 你能顺畅地把其他人必须做的事情委派给他们去做吗？

如果两个问题的答案都是肯定的，那么恭喜你，你已经具有比较好的委派任务的能力了。

如果两个问题的答案中有一个是否定的，那么你就需要培养、练习这种能力。

要想比较好地委派任务，需要做到以下几点。

• 了解项目目标。

• 做好项目任务分解。

• 了解团队成员的技术能力和个人意愿。

• 分配任务时，遵循两方面的原则：既要让某位成员做其擅长的，还要给他一些超出能力范围的、带些挑战的；既要给某位成员他愿意做的任务，也要给他一些他可能不是特别乐意做的任务。

• 以交付为目标，以人人满荷为策略，统合不同成员的任务关系。

要想做好任务委派，必须要合理运用共情、目标统合、辅导、选择、反馈、组织、管理、激励等多种能力。没有谁能够一到管理岗位就能做好任务委派，都是在工作中不断练习才慢慢掌握的。

3）目标统合

作为公司，一定有自己的愿景和目标；作为团队，也有团队的目标；而每一个团队成员，又有自己的目标与诉求。只有这三者一致时，才能产生协力。

目标在从公司高层传递到一线员工的过程中，各个层级的领导者拥有至关重要的作用。他们必须充当合格的解释官，把上一层的目标合理地转化为自己团队的目标，清晰明确地解释给团队成员。唯其如此，团队成员才能明了团队为什么要这么做、他自己又该怎么做。而现实中，有相当多的管理者把信息视为资源，把信息差视为权力的基础，从主观上阻碍信息的透明流动，公司的愿景与目标，往往在经过他们的过滤和解释之后，就发生了微妙但巨大的偏移，如图3-1所示。



图3-1

如果经过的层级较多（三层以上，很多大公司有五层或更多），公司目标到达员工时，往往会从爬衡山变成登恒山，此时团队成员越努力实现他们的目标，距离公司的期望值就越远。

所以，管理者首先是一个目标分解者，要具备与他的领导沟通目标的能力，能够做到将上级目标没有误差地转化为自己团队的目标。只有自己准确理解公司目标，从公司目标合理地分解出团队目标，才可能指导后续的团队执行。

而团队在实现自己的目标时，每个成员的个人目标又会与团队目标产生相互作用。我画了一张图，形象地描述了个人目标与团队目标之间常见的几种关系，如图3-2所示。

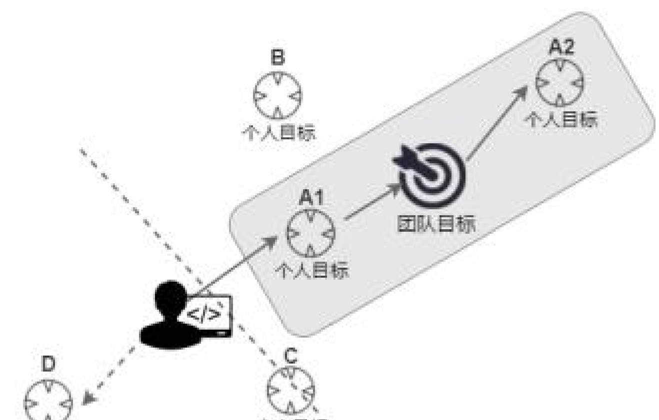


图3-2

如图3-2所示，只有个人目标与团队目标一致时，个人才有最大的意愿去实现团队目标。所以，作为领导者一定要考虑：**实现团队目标，对个人而言意味着什么。**

团队目标对每个个体的意义都不同。所以，领导者要有共情的能力，要能理解下属的痛苦、欢乐、希冀、拒绝，才能针对每个具体的个体来考虑团队目标对于他个人的意义。

所以，领导者的另外一个关键角色就是：**目标解释者。**他一方面要具备解释公司目标、团队目标的能力，能够将团队目标准确地描述出来，另一方面，他又要能共情，能理解每个团队成员的诉求。只有这样，他才能找到团队目标对于每个个体的意义，才能激励每个个体积极实现目标，才能完成目标统合，最终实现公司目标。

而实际情况是，很多管理者往往不加分辨地接受上级丢过来的目标，然后不加解释地压向下级，公司目标在各个层级间都没有得到合理的整合，最后根本不可能实现。

所以，作为管理者，一定要确保团队目标与公司目标一致、个人目标与团队目标一致，只有这样，才能产生协力，有效实现预期。

4）反馈

你在做开发工作时，肯定经常会因为这些情况而感到失落或不被重视：

• 你的领导很少因为你努力工作而对你表达谢意。

• 你的领导很少因为你工作出色表扬你。

• 你的领导很少周期性地与你一对一沟通，帮助你改善工作方法。

你做事时的这些感受，是因为没有及时得到有效的反馈而产生的。

及时地、有效地给予下属反馈是很多技术领导（尤其是刚从执行者转过来的技术领导）都比较欠缺的地方。

技术领导不能给予下属有效反馈，要么因为“自己在组织中拥有了权势，能够左右下属的薪水、奖金，能够对下属进行赏罚，能够名

正言顺地指示下属了”，而想端着架子维持领导范儿不愿意轻易给别人好脸色，要么不知道怎么做好反馈，老表扬怕下属自满，老批评怕下属不满。

所以，最后的情况往往演变为：只要求下属完成任务，不给予及时反馈，等评估绩效时见。所以，很多得到好绩效、拿到奖金的员工其实不知道自己哪里做得好，而很多评级较差的员工则不知道自己哪里有待改善、往哪个方向改善。

每个人都是鲜活的，都有情感上的诉求，他在一家公司工作，绝不仅仅是要充当一部机器干点活拿一份工资，他一定希望得到别人的尊重与肯定，一定希望自己能变得更好。从这一点来讲，管理者应当掌握有效反馈的技能，让员工感受到“温度”和“情感”。

当员工为了上线新版本通宵加班时，哪怕出了问题上线失败，也要适时地表达感谢，抚慰其情绪。当员工出色地完成了工作之后，一定要慷慨地表扬，肯定其做得出色的部分。当员工把事情做砸之后，一定要就事论事，不要评判其人品，要和他一起找到改善的方法，让他知道怎样提升自己。

总之，你要能共情，把员工当作活生生的个体来看待，理解其情感诉求，适时正面地反馈，满足其情感诉求，他才能充满激情地投入工作。

要做好反馈，可以从感谢、建议、评估这三个方面入手（参见《横向领导力》）：

- “感谢”是把你对他人努力工作的感激和赞许之情表达出来。这是一种情感上的表达，目的是满足对方情感上的需要。

•“建议”（或“指导”）是指出你认为对方的哪些具体行为应该坚持，哪些应该改变。此时你的关注点是评价工作而不是评价人。

•“评估”是根据一组明确或默认的标准以及其他人的表现对对方的表现做出评价。

5）辅导，教练式管理

你晋升为管理者后，就拥有了职位附带的一些势力：

•**报酬势力**，比如决定下属调薪，奖金发放等。

•**强制势力**，比如能够对下属进行赏罚，能够决定下属的职位升迁。

•**正当势力**，比如具有指示下属、命令下属的正当权力。

而下属则可能因为畏惧领导的这些势力，而在表面上表现出服从的样子，这会给领导带来一种控制感，当这位管理者不能通过目标整合、共情、反馈等方式让团队成员投入工作时，就会祭出权力的大棒，强制下属执行。

然而，命令和权力是危险的，它们既不能产生信服，也无法形成激励，相反，过度依赖它们还会在团队中形成一股反作用，使得效率和质量下降，阻碍目标的真正实现。

主教练命令国足夺取世界杯，国足就能夺取吗？

下属和团队能否实现目标，不在于命令和强迫，而在于他实现团队目标对他意味着什么，在于他自己有没有要实现目标的自发愿望，在于他有没有自发的行动计划。

当一个人自己找到了目标，导出了行动计划，自发去执行的时候，达成目标的可能性就大大提升了。所以，管理者要做的就是寻找恰当的方式整合团队目标与个人目标，让下属觉得自己做的事情是有价值、有意义、符合他的目标的，让下属觉得他自己也是有价值、被尊重、有选择权的。

为了达到上述的效果，管理者应该尝试放下告知、命令、强迫的管理方式，转向教练式管理。

教练是一种技术，它的目的，是激发自信发掘潜力，帮助下属成长。

教练的方式，通过提问来让团队成员自己明确目标，让他自己导出行动计划，让他用自己的方式达成目标。这样他就感到被尊重，就有选择权、自主性、责任感、动力。

《高绩效教练》一书详细讨论了教练技术，并提出了一个简单易行的GROW模型：

• **目标设定**（Goal），本次教练对话的目标，以及教练的短期目标和长期目标。

• **现状分析**（Reality），探索当前的情况。

• **方案选择**（Options），可供选择的策略或行动方案。

• **该做什么**（What），**何时**（When），**谁做**（Who），**意愿**（Will）。

练习GROW模型，可以先从提问开始。《高绩效教练》也提供了一组有帮助的问题，可以让我们参考：

•“还有什么”，在大多数回答之后使用，会激发更多思考。

•“如果你知道答案，它会是什么？”，它让对方越过障碍向前看。

•“它对于你或是他人造成的结果/影响是什么？”

•“你使用的是什么标准？”

•“对你而言，这件事情最难/具挑战的部分是什么？”

•“如果你的朋友面临你现在的处境，你会给他什么建议？”

•“想象你和你认识或者想象中最智慧的人对话，你认为他会告诉你该怎么做？”

•“我不知道下一步该怎么办，如果是你，你会怎么办？”

•“如果有人对你说/做了这些……你会有怎样的感受/想法/行动？”

在管理中运用教练技术时，要放下评判，多共情，多反馈，多提开放式问题。

当你采用教练技术后，不但能让下属自己成长，还能让下属高效完成工作，你就能较好地完成管理者的两项任务：完成工作、培养下属。

6）选择

当你成了管理者之后，会面临更复杂的情况，既要接受公司的目标、领导的管理，还要管理下属实现团队目标，也要管理好自己的目标和工作，有时还要支持相关部门和人员的各种需求。

你会发现，你突然多了很多事情，突然要应对很多不同层次的人，你没有自我支配的时间了。

假如你不能区分什么人重要、什么任务重要、什么事情紧急，来者不拒，那你就会自陷泥沼，无力挣扎，整天一团忙乱，但总是不出成果，甚至还会在关键时刻掉链子。你付出了更多时间和精力，领导对你却不满意，下属对你充满抱怨，同级也指责你配合不力……

这个时候，选择的能力就变得尤其重要了。你要围绕着你的目标进行选择，识别关键要务，遵循要事优先的原则。

要识别关键要务，还得回到前面谈过的“目标整合”上来，只有你明确了公司的目标、团队的目标、自己的目标，才能围绕着目标进行选择——那些有助于你实现关键目标的事情，就是你的关键要务。

识别了关键要务后，就要勇敢地选择它们，投入人力、时间、精力，优先保证关键要务的执行。与此同时，你就拥有了拒绝的勇气，能够识别出哪些事情该拒绝，哪些事情可以加入排期，哪些事情可以授权给下属去做。

7）承担责任与压力

选择需要勇气，还需要担当——你要能够并且愿意承担你的选择带来的后果。比如有的部门会抱怨你支持不及时配合不力，领导可能会因为你拒绝他的一些临时性安排而认为你不够听话。

所以，你要牢记目标，承担选择的后果，承受来自领导、同级、下属、客户等各方关系人施加给你的压力。

假如你只想享受权力和回报而不愿承担责任，那么你就不配做一个管理者——承担责任和压力是管理者的基本要求。

只有你顶住压力，做好团队的隔离墙，团队才能集中精力处理关键要务，才可能更有效地实现目标。

8）复盘

在上一章——如何在技术上持续精进中，我们已经讨论过复盘应当成为个人习惯。对管理者来讲，复盘的能力尤为重要，只有善用复盘，你的管理能力才能不断提升。

作为管理者，你做得好坏，不再是你一个人的事情。你的一个错误决定，就可能让整个团队的所有努力白费。而要领导团队完成目标，是高难度的任务，管理者必须在每一个项目（或者每一个里程碑）交付后回顾总结：

• 这次项目的目标是什么？

• 现状是什么？

• 现状和目标之间有哪些差距？

• 项目管理哪里做得好？是怎么做的？是否可在类似背景下应用这次的经验？

• 项目管理哪里做得不好？为什么？有什么办法可以改进？

• 团队里谁做得好？好在哪里？他做了什么关键事情导致了好的结果？

• 团队里谁做得不好？哪里不好？是能力、方法、态度等哪方面有问题？改善哪方面可以让他更高效？

• 自己在领导、管理方面，哪些做法取得了好的效果？是怎么做的？是否可在类似背景下应用这次经验？

• 自己在领导、管理方面，哪些做法带来了不良影响（对团队成员、对事情推进、对最终结果、对自己成长）？是能力、方法、态度等哪方面的问题？有什么可以改善的？

• 上司对项目结果满意吗？满意哪些方面？他为什么对这些方面满意？

• 如果上司对结果不满意，是哪些方面让他不满意？他为什么对这些方面不满意？

• 在和上司的沟通协作中，有哪些地方做错了？有哪些地方可以改进？有哪些地方做得好？能不能固化成经验？

学习→实践→复盘→学习→实践→复盘……这是成长的循环，这里面尤为重要的是复盘，只有你带着审视的眼光去看待你做过的事情，一遍一遍深入回顾总结，这些事情才能升华出经验和教训，指导后续的学习和实践。

复盘是经历炼化的关键。很多研发团队，项目一个接一个，节奏快，时间紧，往往是做完一个扔一边，再做完一个再扔一边，丢掉了最佳的成长机会。这是极为可惜的，只做不想、只做不复盘，你成长的速度、团队成长的速度，都会极慢。

项目是一座宝藏，复盘是开启它的钥匙，请好好珍惜。

3.4 走向技术管理的4种方式

当你找准定位、决定要向技术管理转型时，必须考虑一个问题：怎样才能从开发岗位走到管理岗位？

通常来讲，有4种路线：

- 技而优则管。
- 从打杂到管理。
- 从大公司跳入小公司。
- 获取PMP证书，切换到有需要的环境。

接下来我们详细描述这4种路线的常规做法，你可以对照着看看哪种路线更适合自己。

1.技而优则管

曾宪杰2007年6月份加入淘宝技术部，头衔是C++开发工程师。当时淘宝网技术部写C/C++的开发者加上曾宪杰一共三个人，C/C++当时的需求也不大，做了一个小功能后曾宪杰被要求转Java，接手了一个现有功能的改造的任务——把当时淘宝店铺商品分类结构由一层改为两层以及把一个商品只能归属一个分类改为支持多个。那个时候曾宪杰还在在试用期，挺担心自己能不能顺利搞定任务转正……虽然之前鄙视过Java，不过曾宪杰学习能力很强，能够举一反三，很快把Java应用起来，搞定了功能改造任务。

曾宪杰顺利转正后，被安排去研究消息中间件，开始了Java中间件之路，基本上2007年年底到2008年年中，一直负责消息中间件的设计和实现。

消息中间件是淘宝网非常重要的产品，如果挂了，整个淘宝的交易都会出问题。没怎么搞过Java的曾宪杰一上来就直接负责这个产品，可他不但承受住了压力，还觉得这样的开发工作充满了单纯的快乐。

作为曾宪杰自己完成的第一个Java产品，消息中间件上线后运行状况良好，没有出现过什么大问题。他的老板对这个产品的表现非常满意，给予曾宪杰很好的评价。

2008年下半年，曾宪杰的老板安排他搞分布式数据层，这是一个曾宪杰不熟悉、当时淘宝技术部也没多少人懂的东西。从2008年年底到2009年年初，曾宪杰的基本重心就在搞数据层。也是在这一年，曾宪杰开始带人工作，当时带了5个人。

曾宪杰带人把分布式数据层搞出雏形后，老板安排他做同城容灾。同城容灾非常复杂、麻烦，应用间的依赖梳理、依赖的单Active集群的切换、整体容灾测试等，都有不小的难度……

在不断搞定难题的过程中，曾宪杰带领的团队人数越来越多，管理的工作占比也越来越大，慢慢从开发岗位切换到了管理岗位。

2010年1月，曾宪杰成为中间件团队负责人。

2013年5月，曾宪杰升任淘宝技术部总监。

曾宪杰最初担心转正，后来搞定消息中间件，得到认可，开始带人，再后来搞定分布式数据层，在团队中的位置越来越重要，再后来带人搞定更为复杂的同城容灾……他在一路升级打怪的过程中，自然

而然地带人、带团队，最后切换到了技术管理岗位，后来晋升到总监。他走出的技术管理之路非常典型，类比古代的“学而优则仕”，我们可以叫它“技而优则管”。

“技而优则管”一般分为三步：

如果你的技术能力很强，在某些技术方向上有深厚的积累，能够解决复杂的问题，在项目中做出了成绩，那么你往往会被领导注意到，会让你带一些新人或者不如你资深的同事。这是走向技术管理的发端，第一步。

你带人带得不错，小伙伴们跟着你能够快速成长，你就可能被安排带项目，带着你的小伙伴一起做项目，此时你因为技术能力强，又有带人的经验，一些项目管理、人员管理的事情就会由你来完成。这是走向技术管理的第二步。

当你带着一个小团队，完成了项目，并且结果还不错，符合领导的预期时，你就在领导那里留下了“某某可以带团队做管理”的印象。如果你持续带人、带项目，就会不断加强领导心中的这种印象。

最后，若有做管理的机会，你就会顺理成章成为技术经理或研发经理。

同样走“技而优则管”路线的，还有某知名视频网站研发总监老陆。

老陆2000年参加工作，做银行监控系统，接触到视频处理和DirectShow。

后来DirectX 7发布时，DirectShow成为DirectX的主要组成部分。老陆认为这是一项非常棒的技术，就跟进学习、应用，那时使用DirectShow开发多媒体应用的公司很少，国内也没什么一手的资料可

以参考，他就从SDK开始，每一页文档都认真细致地阅读，SDK所有示例的代码，每一行他都认真地阅读并理解，从DirectX 7，到DirectX 8，再到DirectX 9，他一直跟进，实践，积累了丰富的知识和经验。

老陆在工作中应用DirectShow，并且在网络上分享自己的经验，积累了很高的个人声誉，被称为“国内DirectShow第一人”。

2003年，老陆出版了第一本DirectShow专著——《DirectShow开发指南》。

2004年，因为在技术上的出色表现，老陆开始带四、五个同事做开发，2005年，开始做研发经理。

老陆转型技术管理职位，同样是典型的、教科书式的“技而优则管”，这也是技术出色的开发者顺其自然的转型路线。

研发团队的管理者，有一大部分都是这样走过来的。

2.从打杂到管理

很多人觉得软件项目经理或研发经理必然是技术大牛，他们认为：

- 只有技术出色才能服众，才能领导他人。
- 只有技术出色才能更好地确定需求是不是可以实现。
- 只有技术出色才能搞定复杂的问题。

这也是我最开始的想法，认为技术对管理者非常重要，不牛不足以领导开发者。但当我做了管理之后才发现：这种想法是彻头彻尾错误的！

为什么呢？

因为真正决定你是否能做好管理的，并不是技术是否出色，而是你是否能把杂打好，即：

• 能不能准确理解需求并传递给程序员。

• 能不能了解每个人擅长什么、想要什么。

• 能不能合理地安排任务，让每个人既能干擅长的，又能遇到一些挑战。

• 能不能协调各种资源，让程序员可以顺利开展工作。

• 能不能屏蔽高层、需求、产品、市场、售后等相关干系人的干扰，让程序员可以在一个时期专注地做一件事。

• 能不能做好项目计划并跟踪执行。

• 能不能做好团队或部门规划。

• 能不能给程序员创造成长的机会。

上述事情的结果，并不取决于你的技术多厉害，而取决于你是否真心想为大家服务、为项目服务、为公司的目标服务。

很多“技而优则管”的开发者，都会经历一个艰难的角色转换过程：技术越厉害，就越不容易放下技术优势，就越想以“我技术能行”来证明“我管理能行”，就越难渡过从执行角色到管理角色的转换期。这些开发者走过的痛苦旅程，说明技术能力强弱并不能直接决定管理水平的高低。

而意识到这一点的开发者，则可能选择另一条通往技术管理者的道路：打杂。

观察一下所在的团队，盘点一下你在做的项目，有多少开发之外的杂活儿：

- 组织会议
- 讨论需求
- 讨论UI
- 制定开发计划
- 开发计划执行与跟踪
- 开发者状态搜集、更新、同步
- 项目状态更新
- 编译版本
- 送测
- 培训客户
- 培训售前
- 培训运维
- 培训售后
- 为新成员介绍项目

- 培训新成员
- 问题（Bug）搜集

不同模块开发者之前的协调

- 文档撰写

看看你的上司会做哪些，观察他不愿意做哪些，然后，你就可以主动把这些事情揽过来做。

对，主动站出来，主动揽过来！

大部分的开发者因为觉得：

- 麻烦
- 影响自己的技术精进
- 老替领导干事儿影响自己在同伴中的形象

而躲避这些事情。但实际上，这些事情正是一个管理者日常要做的事情，如果你有机会去做，不但替怕麻烦、怕重复的领导（有些管理者做过一两次熟悉之后就不愿再做）分担了工作，还让自己提前得到了锻炼，培养了自己管理、组织、协调、沟通、表达、凝聚力、执行力等各种能力。

更为重要的是，当有新的项目管理的机会时，你会因为已经熟悉这些所谓的“杂事儿”而变成最佳人选！

所以，开发做到一定程度，跳出你的职责范围，从打杂开始，走向技术管理，也是一条现实可行的道路。

我有位朋友，在IBM西安分公司做开发，技术水平在他们那个部门只能排到第4或者第5，但他属于领导眼中什么杂事儿都可以做的人，比如大家下午开会晚了要吃东西订盒饭，就让他打电话；比如领导出差去北京，有事情要安排别的同事做，就让他传口信儿；比如某个产品要发布，现在还有哪些问题，也让他搜集；比如开周会，项目状态、每个人的状态，也让他汇总后发邮件；比如团建吃饭找地方也让他来定……诸如此类，大部分开发者避之唯恐不及的事情，他不但来者不拒，还努力做到让领导满意。

他很明白一点：如果想晋升的几个人实力不相上下，那么最终有晋升机会的，往往是和上司协作更顺畅、上司更了解也更信赖的那个人。

2015年，这位朋友的领导晋升二线，一线经理的位置空出来，他顺利升职，成了新的一线经理。

作为开发者，要想转向管理岗位，如果你的技术能力卓越，远超他人，那么你有极大可能通过“技而优则管”这种方式自然而然地走到管理职位；如果你的技术能力一般，就必须努力通过别的方式让别人（尤其是领导）意识到你是有管理能力的，这种方式就是跳出职责范围去打杂、去为领导分担他不愿做又必须完成的工作。

不管是哪种方式，都符合图3-3所示的模型。

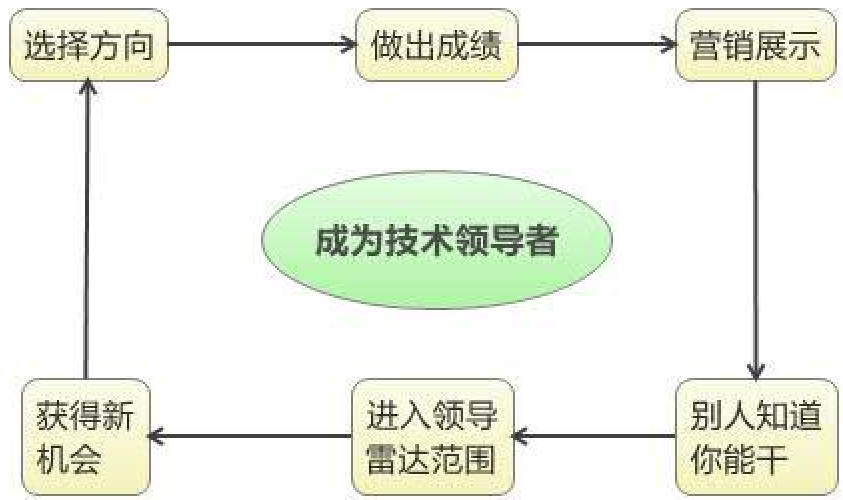


图3-3

在这个模型中，要特别提醒一点：营销展示。因为很多人还秉持着这样一种观点：只要我认真、努力地工作，总有一天领导会发现我，给我想要的机会。但实际上，除非你的优秀超出别人10倍，否则别人很难主动跑过来发现你的特别之处并把机会送给你。

所以我们不但要有效工作，做出成绩，还要展示成绩给别人看。营销展示的方式很多，比如周报、月报，可以让上司知道你都做了什么、取得了什么成绩；比如产品功能演示，在相关干系人了解产品的同时，也可以展现你的演说、表达等能力；比如技术分享，可以让别人知道你在某个方向有丰富的积累……

3．从大公司跳入小公司

在大公司工作，技术能力卓越的人很多，打杂打入化境的人也很多，如果你不是特别优秀或机缘巧妙，就很难有出头之日，想成为管理者非常难。

如果你厌倦了没人注意的凤尾生活，想要在职能上更进一步，切换到管理的跑道上，那么有一个非常诱人的选择：用你现在的大平台

为你背书，跳到小公司做技术管理。

很多初创公司的技术负责人、研发经理、CTO，都是从大公司出来的。

2016年4月份，一位朋友通过在行（在在行APP内搜索“安晓辉”即可找到我）约我，聊研发团队组建的事情。他本人是华为某个产品线的高级工程师，机缘巧合认识了一家医疗背景的公司，该公司要在移动医疗方向创业，想请他负责软件研发工作，通过成立子公司的方式来运作。这家子公司由这位工程师负责成立，医疗公司注资300万元，工程师和他组建的团队可以占一定股份。

和我聊过之后，这位工程师从华为辞职，自己担任子公司的CEO，开始了创业之旅。

这是双创浪潮下，大公司的开发者成为创业公司管理者的实例。

在我们的身边，这样的开发者很多，他们在大公司内没能走上管理岗位，就选择到小公司去担任技术线的负责人，搏一搏明天。

4.获取PMP证书

有一些开发者，技术能力不错，但还没到“放哪里都很突出”的程度，想转型做技术管理，走不了“技而优则管”的道路，也不太想走打杂的路线，就会考虑这条路：参加PMP资格考试，获取证书，切换到对PMP资格有要求的组织中去从事项目管理工作。

有些单位在招聘项目经理时会要求有PMP资格或注明“有PMP资格者优先”。但有多少开发者通过获取PMP证书成功转型到项目管理方向，我这里没有相关数据，身边也没有案例。

这条路有两点需要注意：

• 考取PMP资格证书。

• 在实际开发工作中，把自己放在项目经理的角度上去看待项目，使用PMP的理论、方法、工具等来复盘所做项目的管理过程，提升项目管理能力。

3.5 怎样为成为管理者做准备

要成为技术管理者，你得是一个有准备的人，这样当机会来临时，你才能抓得住、搞得定。

要成为有准备的人，关键就在于：你还没负责某事时，就做好了相关的方案；你还不是某个角色时，就准备了它所需要的能力。

有的开发者是天生的领导者，他们从孩童时代起就已经在培养领导能力了，比如玩游戏时他们制定规则、安排玩法，上小学、中学时担任班委管理班级事务，上大学时又加入社团担任关键职位组织各种活动。他们在这些实践中已经具备了部分管理技能。

而有的开发者（比如我），并不是天生就具有领导能力，甚至有的朋友从小到大都在回避管理方面的事情，尽量让自己不要成为被注意的焦点。这些开发者想要转型管理岗位时，就会面临更大的困难，需要专门学习、训练，让自己具备领导该具备的行为和思维，掌握管理所需的各项技能。

不管是初具领导与管理技能的开发者，还是尚未展现领导才能的开发者，要想成为技术管理者，都需要做进一步的准备，只是前者准备的过程可能更为顺利（因为他们已有相应的意识），后者准备起来遇到的困难会稍多一些。

一个人的能力其实分为两个层面：知识和技能。

所谓知识，就是你知道的、能引发你改变的信息。

举个例子，C++的虚函数就是知识，你学会了虚函数，就学到了这个知识。

所谓技能，就是你运用知识解决实际问题的能力。

延伸虚函数的例子，你能运用虚函数的原理，设计出一个接口，对线段、三角形、正方形、矩形、椭圆形、五角星等形状进行抽象，实现2D绘图，最终完成一个流程图软件的制作，这就是技能。

要为技术管理者做准备，可以从知识和技能两方面着手。

首先，你要储备管理知识，这一点可以通过阅读来完成。其次，你要想办法实践你学到的知识，把各种管理知识内化为你的技能。

学习管理知识，最便捷有效的方法是阅读。

在阅读时，可以先学习纲要，再学习专项。

技术管理者是知识工作者，做的也是知识工作的管理，要学习管理知识和框架，最好的教科书，就是现代管理学之父彼得·德鲁克的著作。像《管理的实践》《卓有成效的管理者》《管理：任务，责任，实践》《巨变时代的管理》《创新与企业家精神》和《21世纪的管理挑战》，都是超级经典书，一路读过来，你就会对现代管理有一些基础性、纲要性的认识。

再回到技术管理的另一个方面——技术领域，在这方面你需要了解技术角度的管理知识和实践，最经典的书，莫过于《人月神话》《成为技术领导者》和《人件》这三本书。

做技术管理，免不了要涉及具体的项目管理，所以你要建立项目管理知识体系，最经典的图书就是《项目管理知识体系指南（PMBOK指南）》。而软件工程方面的知识也是必需的，这方面有两本比肩的巨

著，一本是Lan Sommerville（萨默维尔）所著的《软件工程》，一本是Roger S. Pressman（罗杰S.普莱斯曼）所著的《软件工程：实践者的研究方法》。

涉及具体的软件项目，还有一些类似“设计模式”的项目模式你需要了解，看《项目百态：深入理解软件项目行为模式》这本项目行为模式圣经即可。

当你仔细阅读了上面提到的这些书之后，现代管理学、技术管理、项目管理等方面的知识储备就有了。然后就可以针对管理者所需的专项能力来进行有针对性的阅读，丰富自己的管理技能导向类知识。

比如沟通和说话方面，看《所谓情商高就是会说话》《关键对话》和《内向者沟通圣经》；比如委派任务方面，看《交办的技术：职场晋升第一课》和《别让猴子跳回背上》；比如带人方面，看《带人的技术：不会带人你就自己做到死》；比如目标管理方面，可以看《目标管理实务手册》；比如辅导能力方面，可以看《高绩效教练》；比如演讲、表达方面，看《高效演讲》《金字塔原理》及《演讲的力量》；比如感染力和影响力方面，看《影响力》《横向领导力》《你的团队需要一个会讲故事的人：用故事思维解决问题》和《认同感：用故事包装事实的艺术》；比如时间管理方面，看《小强升职记》和《搞定》；比如高效工作方面，看《高效能人士的七个习惯》和《做事的常识：事情一来，马上就知道怎么做》……

我们在阅读时，一定要结合自己的工作和经验来理解书中的知识，并且反过来用，拿书中的知识、方法、工具来分析你身上、你身边所发生的事情，琢磨事情的好与坏，琢磨如何改善。唯其如此，你学到的知识才能慢慢内化。

当你储备了技术管理所需的知识之后，就要努力去发现工作中管理相关的机会，去应用你所学到的知识，把这些知识慢慢地变成你的技能。

要在工作中发现管理相关的机会，**关键在于你要跳出工作内容和岗位职责的范围，放眼全局来思考。**分三个方面，**第一方面**是产品或项目本身，多想想自己要做的软件到底要解决什么问题，给用户带来什么价值，为什么要做成这个样子，它是怎样被用户使用的；**另一个方面**是工程实践，思考整个项目，从立项、需求、开发、测试、交付、运维、项目生产工具，分析哪个环节做得不好，哪个环节改善之后效果可以被看见，主动去做这些事情，主动成为衔接不同环节、不同人员、不同部门的桥梁；**第三方面**，发现领导的目标是什么，他看重什么，他在为什么困惑，自己能不能帮到他，因为从某种意义上讲，你的工作，就是为了让领导的工作更顺利，你的工作目标，就是实现领导的目标。

“从打杂到管理”一节所举案例中的朋友，就非常明白，义无反顾地跳出职责范围，抓住各种可以做事的机会，锻炼自己管理方面的能力。最终他也因此成为有准备的人，在原来的上司升为二线经理后，自己晋升为一线经理。

在工作中，在还未成为管理者之前培养相关能力，有4种策略。

1.站在项目管理角度看问题

站在项目管理角度看问题，琢磨现在的项目是怎么管理的，有哪些问题，怎么改善。给自己定一个目标，争取每天挑出一个问题来琢磨解决之道。

比如代码管理工具使用svn而不使用git会有什么问题；

比如每日站会上有些程序员每天都说“昨天改Bug，还没查出原因，今天接着改Bug”到底有无意义；

比如开发人员因为技术实现原因而选择忽略App的视频启动动画，需求人员不同意，该怎样和她沟通；

比如前端开发者元小二经常性地把Bug直接分配给后端开发袁大头，两人为此不断口角，怎么解决；

比如有几位程序员，提交代码时不写任何日志；

比如阿三提交代码10次中有8次不做集成构建测试；

……

2.主动承担职责外的杂事儿

主动承担与组织、协调、沟通、管理相关的事情，比如组织会议、跟踪计划、分派任务、推动多干系人协作等。

这方面如果你有心，就会发现很多可以发挥的机会——因为大多数开发者都嫌麻烦，不想把事情揽在自己身上。

比如团队开会时，你每次都可以主动做会议记录，事后发送给大家；

比如你可以帮助预订会议室，协调时间；

比如你做前端，产品经理提了一个需求，需要后端对应也做些更改，你可以不用等产品经理去推动，自己去找对应的后端开发者，商量解决之道；

比如领导每周开会前要汇总团队所有人员的状态，一封邮件一封邮件地整理很麻烦，你可以从群组邮件中搜集信息，把大家的状态汇总成一封邮件发给领导；

比如项目要发布了，还有很多Bug，你可以到jira上汇总开放状态的Bug，根据模块和严重程度分类汇总，提交给项目经理或者分享给大家；

比如部门要招人，你可以主动请求帮助经理来筛选简历，设计笔试题目，批阅笔试卷子；

……

3.设想你是技术管理角色

站在管理者的角度思考，向身边的管理者学习，复盘身边的管理者，看他们哪里做得好、怎么借鉴，哪里做得不好、怎么改进，设想如果自己是管理者会怎么做。

比如大家觉得项目经理很愚蠢，不考虑需求A技术上能不能实现就直接答应客户，那么你是否可以琢磨一下，他是在什么情境下答应客户的、可能出于什么考虑，然后把自己放在他的位置上，如果你是项目经理，在那样的情境下，会怎么做？

比如你觉得团队里的武二郎一天到晚打游戏、看小说、刷新闻，工作时间经常不足1小时，可是经理一直都没对此做过什么事儿，那么你把自己放在经理的位置上，想想要怎么处理这种情况。

比如你发现团队气氛消极，大家都没什么往前冲的劲头，Redmine上的任务虽然排好了日程，可是都不在意，今天搞不完就明天搞，交付期过了就延期，从来也不会为了某个子任务的交付而主动加班加点，经理每次开会都强调每个人都要对自己的工作负责，可是几乎没

什么用，开会时大家沉默不语，散会后照旧，如果你是经理，你会怎么做？

4.找一位职业楷模

假如你想成功地走向技术管理岗位，自己学、自己琢磨是一方面，另外还有一个非常重要的提升途径：在组织内找一个从事技术管理岗位的人作为自己的职业楷模。

寻找这样的职业楷模时，要考虑他实际的管理水平、团队的状态、他个人的行事风格是否与你的价值观相符。他一定要是你想要成为的那种人。

如果你的直接上司刚好就是你理想的榜样，那是最幸运的，你可以近距离观察他如何做事，向他学习。

如果你的直接上司不是你的职业楷模，也没关系，你可以在公司范围内去找。一家公司能运转得不错，一定有非常优秀的人在发挥重要而关键的作用，不在你的部门，就在别的部门。所以，一开始找不到也不要着急，就先自己学、自己琢磨，与此同时，跳出你的职责范围，跳出你所在团队，主动结识、了解其他人，当你了解了更多人以后，总会找到的。

一旦你找到榜样，就可以主动结识他，向他学习。可以有几种学习方式：

- 请他做你的导师，请他指点你、辅导你。
- 三不五时地找他聊聊天。
- 观察了解他的事情，复盘他的管理行为。

优选第1种方式，如果对方不愿意，选择第2种，如果第2种也不可得，就用第3种。

在工作之外——生活中，也有很多锻炼管理能力的机会：

- 参加领导力培训课程或者训练营等，跟着老师的设计训练自己，和一群人一起提高。

- 把自己的事情作为项目来管理，比如装修、旅游，都是很好的项目管理机会。

- 组织活动，比如春游、读书会、同学聚会、主题沙龙等，可以锻炼领导、管理、组织、协调、沟通等各种能力。

不论是在工作中还是在生活中，只要你记得自己的目标，就可以发现很多练习的机会，只要你主动把握每一个机会来练习、提升，你就能慢慢地为成为技术管理者做好准备。

在准备的过程中，有两点非常关键：

- 用心发现那些衔接不同环节的、需要有人组织协调的工作。

- 站出来，主动承担责任。

第4章 技术管理新人面临的挑战

在“成为技术管理者”一章中，我们看到，开发者走向技术管理岗位，是从自己做事转变为通过他人完成工作，是从执行角色转变为管理角色，是职能上的转型。这种转型，是从管理自我到管理他人的转变，管理自我所需要的能力与管理他人差异巨大，这种巨大的差异，往往会给刚转型来的技术管理新人带来严峻的挑战。

本章会介绍技术管理新人经常遇到的18种挑战：

1）角色转变

2）被动管理

3）弄不清职责

4）委派任务

5）目标管理

6）资源管理

7）压力管理

8）冲突管理

9）绩效变差

10）担心失去技术竞争力

11）有效的反馈机制

12）别人的议论

13）和下属进行一对一沟通

14）怕犯错

15）时间管理（领导者时间被拆分为5份）

16）激励他人

17）向上管理

18）提升领导力和管理能力

4.1 挑战1：角色转变

从技术岗位晋升而来的管理者，往往是某个技术领域的优秀者，具有很强的执行力和解决问题的能力，在他们刚开始做管理工作时，往往会顶着管理者的头衔做执行者的事情，成为一个“super-doer”。

因为他们在技术上很厉害，可能经常会拿自己的技术水平衡量团队的其他人，觉得这个任务张三很难处理好，那个任务李四铁定犯错误，于是不放心把事情交给别人来做，或者交给别人做了又因为看到要出错，忍不住自己伸手去做，把分给团队成员的任务再拿回来自己做。

当一个技术领导因为担心下属会出错或不能按自己预期完成任务而收回这个任务自己做时，要么会让下属自己觉得自己无能（或者让下属猜测领导认为自己无能），要么会让下属觉得这个领导越俎代庖不干他该干的事，这就会产生严重的不良影响，不利于团队成员自己成长、自己解决问题。同时，这位技术领导也会因为过分关注技术细节而忽略其他的组织、领导工作，导致“只见树木，不见森林”，严重影响整个团队的效率和生产率。

当你作为开发者时，是个人贡献者，管理好自我即可，你有过硬的专业技能和到位的职业意识，可以积极追求并实现个人的高绩效。但你一旦成了技术管理者，角色就变了，除管理自我之外，你还要管理他人。你的工作是通过他人完成的，一定要重视管理工作，而不是凡事亲力亲为（不管出自什么原因）。

团队更需要你做规划，定目标，跟踪计划，协调资源，他人更需要你辅导而非代替他们做事情，甚至你告诉他们怎么做他们也会厌烦

——因为每一个合格的开发者都希望自己搞明白怎么做，都希望自己搞定而不是成为你的某一只手，时刻被你控制。你也许更擅长发现并解决问题，你看着他们找不到方向会难以克制替他们解决的冲动，但你依然应该袖手旁观，让下属自己搞定——顶多给予指导。这样他们才能成长。

谨记，一定要把更多的精力放在人、流程、项目上，你的工作是保障别人的工作能够顺利开展，是创建一个可以让大家各尽所能、实现团队目标和个人成长的环境。

要顺利完成角色转变，下列事情可能有帮助：

- **和你信任的同级别小伙伴多聊聊**，看看他们每天的时间是怎么安排的，他们是怎么看待他们的工作的，你可以从他们那里获得启发、方法、工具，然后结合自己的具体情况，看看哪些适用、哪些不适用，做做实验（还记得“如何在技术上持续精进”那一章介绍过的“对标管理法”吗？这里也用得上）。

- **阅读**。多读领导力和管理方面的书籍，很多前人的经验都能在这里找到，我们在上一章也列出了很多书籍。

- **培训**。有一个很奇怪的现象，很多开发者会希望参加某种技能培训，比如Qt开发培训（需要的话联系我好了），比如机器学习的培训，但是当一个开发者被晋升为管理者时，他却很少想要去接受领导力和管理技能的培训——似乎管理是自然而然就会的事情。不但开发者这么做，很多中小型公司的高层管理者也在这么做：把一个不懂管理的技术人员提拔到管理岗位，不提供任何培训就希望他们能做好管理工作。这是多么令人费解的事情啊！正确的做法是，在成为管理者之前就接受管理方面的培训，走上管理岗位后要接受管理培训，做了一段时间管理工作后还要接受管理培训，即：**根据你所处阶段，持续接受适合你状况的领导力和管理技能培训。**

- **导师。**你需要一位更理解研发团队管理的人来做你的导师，在你遇到问题时可以向他请教，你还可以周期性地和他沟通，从他那里获得反馈和指导，这样你会以更大的加速度前进，快速完成从执行到管理的转变。导师必须是你充分信任的且和你没有直接利害冲突的人。

4.2 挑战2：被动管理

刚上任的技术管理者往往还习惯于做具体的事，把大部分精力放在设计、编码、解Bug等具体工作上，只留少部分时间和精力给管理，甚至会认为管理岗位没什么可做的，有问题了才需要管，没问题则不用管。

这种被问题驱动的管理方式，就是被动管理（消极管理），和故障驱动式开发（开发工作被迫围着故障开展）类似。

假定你安排了一个模块给袁大头，要求他两周后（2017年8月10日）交付，然后你就写自己的代码去了。等到8月10日，你想起袁大头的任务该提交了，就过去问他："怎么样，代码提交了没？明天要联调。"袁大头看看你，说："没做完，估计还得一周。"于是你生气了，指责他工作不积极主动、不负责任、明知道做不完也不想办法赶进度。于是你责令袁大头在接下来的几天里每天晚上加班，周六、周日加班，必须赶在8月14日提交。袁大头满腹怨言地开始工作……

这就是被动管理。你在一开始并没有做计划，也没有风险评估和备案，开发过程中也没有定期跟踪任务状态，更没有根据袁大头的工作状态调整计划，只是到了交付这一天，验收时发现延期，于是被动地安排加班赶进度，这样你、袁大头、测试、产品等相关人员，都因此而陷入了被动，都不得不被"袁大头任务延期"这个问题牵着走。

周一（2017年8月28日）下午开完周会，袁大头找你说："老大，有没有时间？我想找你聊聊。"你满腹疑惑地找了个会议室，问袁大头什么事情。袁大头说："老大，我准备离职。"你大惊，赶忙说："哎呀大头，怎么回事儿啊，不是干得好好的嘛，为什么突然要离职？"袁大头说，"老大，那边催得很紧，我已经答应他们9月11日入

职了，你看这两周能不能安排个人和我交接。”你更惊：“你为什么要离职啊？再说这时间也太紧了，一时半会儿不好找人接替你的工作，你看能不能再考虑一下？”……

这也是被动管理。“一直都好好的”，突然你就碰上了“袁大头离职”这个问题，不得不找人接替他的工作，不得不向上级请示，不得不安排招聘，不得不向袁大头所属项目的相关干系人解释……

被动管理对个人、对团队、对公司，都有百害而无一利。

管理者应该以积极、主动的态度实施管理。

对一个项目，应该在前期花费更多的资源，明确任务的目标、资源、时间、反馈机制、沟通方式、风险，制定相应的计划和应急预案，同时在实施计划的过程中周期性地采集状态，根据项目状态动态调整计划，及早解决各种问题，确保所有项目参与者和干系人步调一致，最终顺利交付。

对一个人，应该在工作中经常性地和他沟通，无论是工作上还是生活上的事情，都要有所了解，要了解他为什么在这里工作，要帮助他制定成长计划，要让他感到自己是特别的、被重视的，让他愿意在这里工作。

4.3 挑战3：弄不清职责

很多从一线晋升的技术领导，一开始不理解经理这个岗位的职责，不知道具体要做什么、怎么做，也不知道公司对该岗位的考核指标、上级领导对这个岗位的期望，这些都是问题。虽然有些公司明确规定项目经理、部门经理等的岗位职责，然而没做过的人，看那些毫无生气的官方描述也是挺让人头疼的，看着都是汉字，每个字都认识，但看了以后就是不知道、不明白什么意思，和没看差不多。更何况，很多公司其实并没有这个说明，或者根本就是从网上或别的公司抄来的，是否适用都没人管。

比方说你看到项目经理的职责里写了这么一条：

确保项目目标的实现，领导项目团队准时、优质地完成全部工作。

对你有实质性帮助吗？再比如下面这条：

与客户沟通，了解项目的整体需求。并与客户保持一定的联系，即时反馈阶段性的成果，和即时更改客户提出的合理需求。

对你有实质性帮助吗？

即便你通过公司的文档了解了岗位职责，对工作范畴有了大概的认识，仍然还是会迷惘：**具体我该做哪些？做到什么程度有没有标准？哪些轻？哪些重？哪些是考核的内容？哪些对我的绩效考核影响大？哪些事情我可以决定？哪些事情必须请示上级？**

问题太多了。你知道作为经理要和客户沟通，然而这并没有什么用，并不能将你眼前的铺天盖地的未知揭开，你只有慢慢去试才会知道水有多深。你是从一个工兵的角色忽然就变成了排长，以前的经验几乎没用了，你还没有掌握新的关于项目管理和人员管理的经验，就必须面对那些事情了，这是一个“负位”的过程，你得自个儿慢慢摸着石头过河去适应。

（注：人们的实际能力，往往低于他所坐的位置，也就是说，有负于他所坐的位置。这种现象被称为负位。）

要想尽快搞明白你的职责，下面几个策略可以帮到你：

• **与你信任的、有经验的其他同级经理多聊聊。**遇到不知道要不要请示领导的事情，遇到不知道哪些该做、哪些不该做的事情，遇到该做而不知道如何做的事情，遇到需要在多个部门间协调而你不知道怎么做的事情……都可以向他们取经。

• **和你的上司多沟通。**你的上司应该是你有力的后盾，应该为你服务。像哪些事情你可以自己决定、哪些事情必须请示他、哪些组织内的资源可以使用、某件复杂事情的影响怎么评估等，你都可以请教你的上司。当然有些不言自明就该你决断的事情，千万不要扔给你的领导，那会让他觉得你职责不清、能力可疑。比如某位程序员没有按时交付、接口API到底是由李四还是王五提供这类事情，你自己搞定就好。

• **和支持部门（人事、行政、财务等）多沟通。**你会有很多牵涉到流程和制度方面的事情要处理，比如下属转正、职级晋升、招聘、辞职、请假、绩效评估、出差、报销、活动经费、团队建设等，假如你不知道怎么做或者不清楚流程，就向对口部门了解。他们会比你的同级同事、你的上司更清楚。

4.4 挑战4：委派任务

我们在“技术管理需要什么能力”那一节已经介绍过“委派任务”这种能力，还介绍说，要想比较好地委派任务，需要做到以下几点。

• 了解项目目标。

• 做好项目任务分解。

• 了解团队成员的技术能力和个人意愿。

• 分配任务时，遵循两方面的原则，既要让某位成员做其擅长的，还要给他一些超出能力范围带些挑战的；既要给某位成员他愿意做的任务，也要给他一些他可能不是特别乐意做的任务。

• 以交付为目标，以人人满荷为策略，统合不同成员的任务关系。

但在实际工作中，即便了解了上面几点，技术管理新人往往还是很难把任务顺畅地委派给别人。这是因为，除了基本的原则，还有很多情感上、习惯上的因素需要考虑：

• 和团队其他成员的关系，从同级变为上下级，不知道怎样对待才更自然，分派任务时担心引起异议。

• 之前习惯了接受别人安排，没有练习过如何分配任务。

• 担心他人做不好，不愿授权。

• 担心项目整体进度，倾向于让每个人做最熟悉的事情，忽略别人的意愿。

• 倾向于给能干的、愿意配合的人安排更多的任务。

经常性地做做心态调试，可以帮助我们更顺畅地委派任务：

• 管理和执行，只是分工的不同，你不认为管理角色比执行角色高级，你不因此不自然，别人也不会不自然。很多时候，你以为“别人觉得你分派工作给他很别扭”只是你这么以为，别人可能无所谓，因为谁来分派工作都是分派工作，都是为了整体目标的实现。每个人都有自己的课题，也都会慢慢接纳他的课题。你的课题就是委派任务，开发者的课题之一就是接受并完成任务。

• 你担心某个人做不好某个任务，这是正常的，但如果你因此而不给他机会，他就一直是做不好的那个人，一直是你担心的那个人。所以你一定要给他机会，只有你给他机会，他才可能做好。

• 每个人都应该满荷工作，而不仅仅是能干的、积极配合的人。如果不断给能干的人更多的任务，迟早有一天，他会因为干不动和感到不公平而离开团队。

• 你不必也不能让所有人满意，只要让大多数人满意即可。

• 不要把臆想当作事实，如果你担心某位开发者对分派给他的工作不满意，就尽早去和他沟通，行动是缓解担忧的良药。

• 你刚开始做管理工作时，有一段时间遇到各种困难，有一段时间做不好（3~6个月），这是正常的，要接纳自己。

4.5 挑战5：目标管理

我们在第3章的“技术管理需要什么能力”一节介绍管理者所必需的能力——目标统合——时提到了三种目标：公司目标、团队目标和个人目标。三者的关系如图4-1所示。

作为技术管理者，首先要和自己的上级沟通团队的目标，尽量使其符合SMART原则（参考第2章“目标的设定与执行”一节），然后再把团队目标拆解到每个人身上，形成个人目标。个人目标也必须符合SMART原则，这样才可能有效执行。

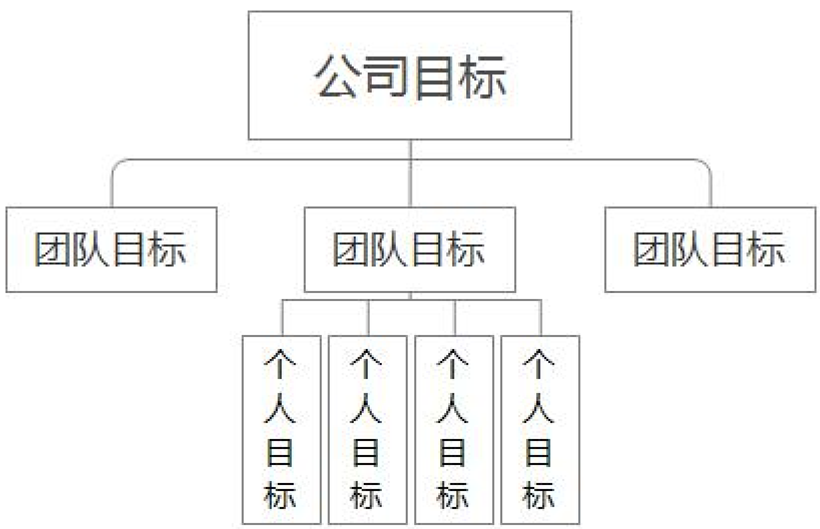


图4-1

技术管理新人在两个目标拆解的阶段比较容易犯的错误就是：

- 搞不清做这件事的意义。
- 制定一个无效目标。

有个客户提了一个需求，要一个纯Web的音视频通信组件，不允许安装Flash、ActiveX等插件。领导找到你说，研究一下看看能不能实现，你问领导："做成什么样子？"领导说："客户也没说，你们先摸索一下。"你又问领导："什么时候要结果？"领导说："尽快吧，不知道客户哪天突然就要了。"然后呢，你就接受了这个任务！

这算什么任务？不知道要做成什么样，违反了S（Specific）原则；尽快交付，没有具体时间，违反了T（Time-bound）原则。

可是你还要把它委派给开发人员！

你简单地想了想任务，做了些信息检索工作，发现H5和WebRTC可以满足不装插件、纯Web实现这两个要求，心中大定。

然后你找到袁大头说："大头，有个事儿你看一下，研究一下怎么用H5和WebRTC做Web音视聊天。"

袁大头问："突然做这个干嘛？"

你说："我也不知道，领导让弄的。"

袁大头问："领导为什么要弄这个？我正做Mac版本呢。"

你说："你先研究一下看看怎么回事儿，搞个DEMO出来。"

袁大头问："那我手上的Mac版本怎么办？"

你说："先放放，你先研究一下WebRTC。"

袁大头又问："做成什么样子？有没有原型？"

你说："没定呢，你先研究一下，看看技术上有没有问题。"

袁大头又问：“要验证哪些问题？”

你有点不耐烦：“哎呀就是试试可行性，看看有没有什么技术难题，你自己先琢磨一下。”

袁大头又问：“什么时候要？”

你说：“尽快。”

袁大头满脸疑惑地研究去了。

这样的场景熟悉吗？

你拆解的这个目标，符合SMART原则吗？（请试试找出来它违反了哪些原则。）

袁大头正在做的任务，被重新计划了吗？（请找张纸写下袁大头的Mac版本该怎么处理。）

这样的目标拆解和管理，往往会打消人的积极性，让人觉得自己的工作毫无价值和意义，同时也让人觉得自己本身不被尊重（袁大头都不知道为什么要做这样的研究）。长此以往，团队的士气、战斗力、协作、产出率，都会下降。

要想摆脱这种状况，可以参考下面的策略：

1）从上级那里接受目标时，要了解目标的意义所在，要努力让目标符合SMART原则。

2）如果接受的目标不符合SMART原则，就自己做一些前期工作，让目标符合SMART原则，再找上级确认。比如示例中，你可以自己找一个别人家网页上的聊天功能，作为DEMO的样子，你可以自己设定一个预研交付时间。

3）分解目标给开发者时，要阐述为什么做这件事，要给出符合SMART原则的目标。

4）如果开发者手头有其他任务，要重新安排该任务的计划。

4.6 挑战6：资源管理

管理者的一个重要工作是保证下属能够开展工作。要做到这一点，就要合理协调各种资源，比如电脑、手机、平板、显示器、路由器、打印机、服务器、BETA环境、VPN账号等，不要让任何资源阻碍开发者开始工作。

我曾经在一家公司工作过一段时间，公司有一款老产品需要支持Mac OS X，要求一位开发者移植这款产品到Mac OS X下，可是公司没有可以给这位开发者用的iMac，也没有Mac Mini，也没有MacBook，经理就去找副总，说申请一台Mac Mini来做开发和测试，副总的回答是——“先装个黑苹果试试，Mac Mini等下个月再买”，结果一等等了三个月，副总也没有协调来一台Mac Mini，那位程序员装了三天黑苹果也没用起来。

关于开发或者测试所用的资源，协调不来往往有客观原因——没钱、要审批、等待发货，等等，但如果管理者不想办法尽快解决，不但会产生很多隐性浪费（其代价可能和买一个设备差不多。比如一个月薪2万元的开发人员装三天黑苹果，时间成本折算成薪水，就和一台Mac Mini差不多了，假如有多个人等在一台设备上，浪费就会更多），还会大大打击开发者的积极性。

2013年我还在信利软件工作，当时有一款产品需要在iPad上测试，公司没有设备，研发人员问我怎么办，我就把自用的iPad拿到单位充当测试机，然后找我们的产品经理，从他那里又借了一台iPad用于测试。与此同时申请购买2部iPhone用于测试。

实体资源的问题，我们总可以想到办法来解决，比如：

• 申请购买。

• 公司内部不同部门间协调借用。

• 借用私人设备。

还有一种无形的资源——开发者的时间，也需要好好管理。这方面需要提醒几点：

• 不要把开发者的时间排满，到70%就可以了。

• 安排工作时不要把加班时间计算在内。

• 要遵循串行工作的原则，不要给一个开发者同时安排多个任务。

• 如果必须给一个开发者安排多个任务，也请做好时间划分，这周做这个，下周做这那个，或者这三天做这个，那三天做那个，尽量不要这个小时做A，下一个小时做B，工作状态保存和情境切换非常耗时，每次切换，也会经历工作效率爬坡，导致整体工作效率低下。

4.7 挑战7：压力管理

心理学家Richard S. Lazarus（拉扎勒斯）提出：压力是由于事件和责任超出个人应对能力范围时所产生的焦虑状态（紧张状态）。

当一位开发者刚刚走上管理岗位时，极容易产生压力——原来他只需要管理好自己，搞定开发任务即可，现在他不但要搞定自己，还要管理下属，还要与产品、需求、测试、运维、销售等部门打交道，还要直接面对更高级的管理，更要命的是，他必须得对整个团队的结果负责任。

责任即压力。

新任技术领导还会担心自己不称职、担忧下属对自己的看法、猜测领导对自己的评价、臆想平级部门对自己的看法，这些心理和情绪活动，会带来连绵不断的压力。

压力会促使人做出某种行为来宣泄。但是在压力下做出的行为，往往是不恰当的。当你是一个管理者时，你一丁点的不恰当行为都会被放大——因为你已经成了团队里的大猩猩，人人都在盯着你看。

比如你只要有一次当着大家的面对某位程序员大喊大叫，大家就会认为你总是爱当着大家伙的面训斥人，丝毫不讲方式、不顾别人感受；再比如你只要有一次迟到（压力过大会导致迟到），大家就会认为你总是迟到……

作为管理者，你一定要时常审视自己在压力下的行为是否恰当，因为你是团队里的大猩猩，你的行为会引发类似的行为。

你在感受到压力时大喊大叫、推卸责任、逃避、沮丧，你的团队慢慢也会变成这个样子。

所以，管理者要尽量避免用消极的行为应对压力，要努力积极地面对压力，勇于承担责任，一切以解决问题为目标。

这是非常艰难的挑战。

不过也有方法，你可以从两方面进行压力管理：

- 换个角度看问题

- 宣泄

心理学大师阿尔伯特·埃利斯1955年提出了著名的ABC原理，可以用于管理压力。因为有时你的压力，是由你对事件的特定解释（B）产生的，只要你换一种解释，压力就会消失，甚至会成为动力。这也就是我们说的“换个角度看问题”。

比如阿金的上司怒气冲冲地要求他解释版本发布为什么延期，并要求阿金给出新的发布时间，绝对不能再延期的发布时间。这时阿金可能会恐慌、内疚、生下属的气，等阿金从上司的办公室出来，马上就把一众开发人员叫到会议室，大发雷霆，质问大家为什么没有按时交付代码。

阿金这种行为恰当吗？

你遇到过像阿金这样“在领导那里受了训教，转过身就训斥下属”的研发经理吗？

那么阿金怎样才能释放掉自己的压力呢？

在这个例子中，

• A是：阿金的上司训斥阿金没有按时发布版本并要求阿金给出确切的发布时间。

• C是：阿金感到恐慌、内疚、生下属的气，然后团队开会，训斥大家不按时交付。

阿金之所以会有C这种不当行为，是因为他第一时间对A做了解释：这帮开发人员不负责任，导致领导认为我不负责任、能力不行，对我失去信心了，如果我不尽快搞定，我以后的工作会更难做。

如果阿金换一种解释：领导生气是对这件事的结果生气，因为没有按时发布，让领导失信于客户，损害了公司的名誉。

用这种解释，阿金接下来的行为（C）就可能改变：对领导的生气表示理解，向领导道歉，然后召集开发人员开会，和大家一起研究项目数据，请大家分析延期的原因，讨论改进的策略，重新制定计划，给出新的发布日期，并取得大家的承诺，然后给领导承诺。

这样的C就是恰当的、积极的行为。

然而作为新任的管理者，当你感受到来自上级、客户、项目干系人等的压力时，下意识产生的反应往往是消极的，要么是在推卸责任，要么是想自我保护。比如这是程序员某某的问题、这是测试人员没测到、这是需求不明确、这是机房停电导致的……然后就开始找某某分析责任……然后就可能陷入扯皮，搞得双方不愉快……

怎么避免这些下意识的不良反应呢？

学会暂停。

当领导批评你、同事指责你、下属埋怨你、产品经理说你的软件没有按需求实现、客户要求你……时，你都先暂停10秒钟再来反应，

避免被打或者逃的本能控制。

电视剧《武林外传》中的郭芙蓉通过一句话——“世界如此美妙，我却如此暴躁，这样不好不好”——来控制自己的暴躁反应，这是一种仪式。仪式化可以帮助我们控制自己的情绪。

你也可以建立自己的仪式，比如默念三声“解决问题”，或者深呼吸三次，或者闭眼10秒钟。

在最开始练习时，你可以暂停久一些，问问自己下列问题：

- 真正的问题是什么？
- 自己这样做，有助于问题解决吗？
- 自己的目标是什么？
- 自己这样做，能实现目标吗？
- 这会让自己现在或将来惹祸上身吗？
- 这会弄僵自己与他人的关系吗？
- 这会给别人带来伤害吗？

当你能够暂停了，这些问题就会瞬间闪过你的脑海，引导你做出恰当的反应。

有些压力换个角度解释一下会消失，有些压力其实会一直堆积在你身上，比如“996”这种长时间超负荷的工作，比如领导行事风格与你迥异，平均每天刺激你三次……这类缓慢的、持续性的压力，必须要合理地宣泄，否则一旦堆积久了，容易导致爆发性的破坏行为。

宣泄应当以不伤害他人为原则。

常见的宣泄方式有：

- **倾诉**。可以向家人、朋友、同事倾诉；可以向专业人士（如心理咨询师）、陌生人倾诉；也可以书面倾诉——写压力日记。

- **声音**。比如大笑，长啸，大吼。

- **哭**。哭是最好的宣泄。想想小孩子，有什么不开心的事情，哭一哭就过去了。男人会觉得哭比较难接受，因为受“男儿有泪不轻弹”的毒害太久了。但是，有什么关系呢？健康更重要。

- **运动**。散步、跑步、爬山，都成，找到适合你的运动方式吧。

- **睡觉**。睡一觉就好了。

4.8 挑战8：冲突管理

（C1）开会讨论网站的技术方案，张三坚持用Node.js+Express+MongoDB+ AngularJS+Bootstrap，李四一定要用经典的LNMP……

（C2）有一个Bug，前端的王五觉得是后端的问题，后端的赵六觉得是前端的问题，两人争执不下，谁也不愿意修复……

（C3）阿金发现袁大头进度落后，要求他加班赶赶，袁大头没办法，答应加班……

（C4）孙八因为技术原因否定了一个需求，可产品经理认为这个效果会严重影响用户体验，必须要做……

（C5）钱久和卢十三在办公室开骂战……

（C6）领导安排你们部门暂停手上即将发布的项目，研究新的项目，你在情感上不能接受，想等手上项目发布后再开始新项目，领导向你解释了新项目对公司眼下的重要意义，并告知了其紧迫性，你衡量后同意了……

（C7）线上的直播流媒体系统出了一个崩溃Bug，无规律，一天一两次，王飞负责修复这个Bug，他在一段代码里加了一个空指针判断，测试了两天，问题不再出现。作为经理的你认为他没有找到真正的原因，希望他继续排查问题，从根源上解决……

类似这样的冲突，研发团队中天天都在发生，作为管理者，你会怎么处理？

在《你好哇，程序员》一书中我记录了两次我经历的冲突，说说其中的一次。

（C8）当时我在一个车载影音项目中，该项目用Windows CE系统，实现DVD、多媒体、收音机、导航等基本功能，另外还有写流式驱动控制马达和电机来翻转屏幕（一开始设备是收在汽车中控盘面板里的，想象一下光驱吧）。当设备唤醒后，驱动通过I2C控制马达，先把机子送出来，然后屏幕上翻，再然后就可以通过蓝牙或者手指触摸来控制设备。

我当时是普通开发者，不是管理者，做界面和流式驱动，而Windows CE的SDK的一部分由另一位开发者（JF）负责。

项目很急，客户急着要去参加CES电子展，我和硬件工程师不久就要现场开发，而驱动还没搞定，我就去找JF，问他什么时候可以联调。

（你懂的，联调可能很花时间，有时各自写代码花不了两天，一集成一联调就完蛋了，写两天代码联调一星期的事儿都可能发生，真是不调不知道，一调死翘翘，这也是敏捷为什么强调持续集成的一个原因，大爆炸式的广度集成，会带来意想不到的悲剧。）

冲突就源自接口联调。

我找JF联调，JF当时不知道在忙什么，不搭理我啊，头也不抬……我就对他说很急，JF回头表示他没时间弄……可是我真的不知道他在搞什么嘛，我就强调项目的急迫性……

后来言辞不合发生了很激烈的冲突，影响极其恶劣，领导分别找我们谈话，了解原委，后来JF被调到另一个部门，再后来JF离职了。再再后来，我也离职了……

假如你是管理者，你会怎么处理团队中的冲突？

托马斯-基尔曼冲突模型（2003年提出），是世界领先的冲突解决方法，它从坚持度和合作度两个方向出发，划分了5种常见的冲突处理方式：竞争、回避、退让、妥协和合作，如图4-2所示。

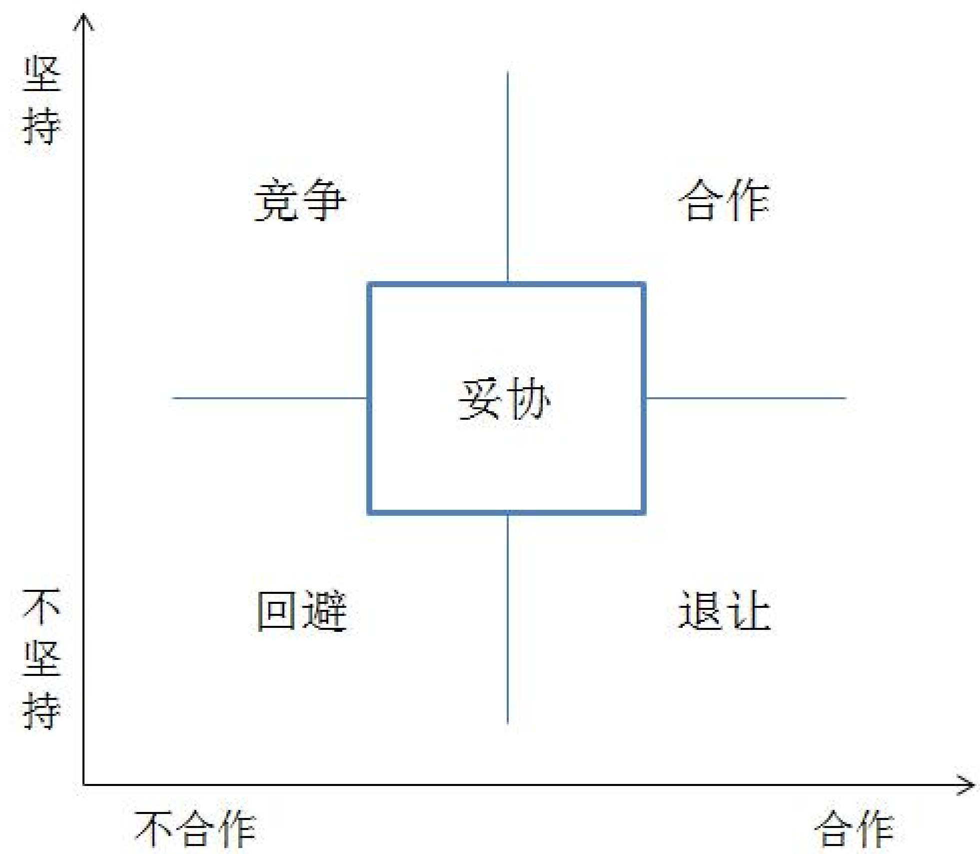


图4-2

我们先简要解释一下5种冲突处理方式。

• **竞争：**高度坚持且不合作，又称为强迫策略，指的是牺牲一部分成员的利益，换取自己的利益或是团队整体的利益，其特征是正面冲突，直接发生争论、争吵，或其他形式的对抗，为了取胜不惜任何代价。

• **回避**：不坚持也不合作，冲突双方意识到冲突的存在，但试图忽略和放弃冲突，不采取任何措施与对方合作，也不维护自身利益，希望一躲了之。

• **退让**：不坚持且保持合作，指一方愿意把对方的要求和利益放在自己的要求和利益之上，做出自我牺牲，使对方达到目标，从而维持相互友好的关系。

• **妥协**：中等程度的合作，中等程度的坚持。冲突双方都让出一部分要求和利益，但同时也保存一部分要求和利益。其特点是没有明显的赢家和输家，他们愿意共同承担冲突问题，并接受一种双方都达不到彻底满足的解决方案。冲突双方的基本目标能达成，相互之间的关系也能维持良好，冲突能得到暂时解决，但也有可能留下了下一次冲突的隐患。

• **合作**：高度坚持且高度合作。冲突双方既考虑和维护自己的要求和利益，又充分考虑和维护对方的利益，尽可能地使双方的利益都达到最大化，最终达成共识。合作方式的特点是冲突双方相互尊重与信任，对于自己和他人的利益都给予高度关注，坦率沟通，澄清差异，致力于双赢。合作的方式能使冲突得到完全消除。

现在我们从管理者的角度，结合事情的重要性和紧迫性来看看，在什么情况下应该采用什么冲突解决策略。

注意，如果管理者不是冲突双方之一（比如C1），这时管理者要解决冲突，矛盾关系就会因为管理者的介入而发生演化：

• 管理者站在某一方的立场上，通过说服或者强迫另一方来解决冲突。

•管理者不站在任何一方的立场上，引导冲突双方自己采取一种冲突解决方式。

•管理者不同意原来冲突双方的办法，提出了新的方案，冲突就变成多方。管理者要采用竞争（强制）策略解决冲突时，就会同时强迫原来的两方。

当事情又重要又紧急，必须快速做出决策时，管理者可以采用竞争策略，强制性地解决冲突。

像C2这种冲突，假如这个Bug紧迫性很高，当天必须修复，凌晨就要上线。那么经理可以采用竞争策略（强制策略），强制王五从前端解决（因为这样更快，也更易于测试和更新）。

像C1这类冲突，假如张三和李四都只是为了技术理想坚持己见而不考虑团队现实情况，作为技术管理者，你知道再这样一周一周讨论下去，项目的开发时间就会非常紧迫，甚至会错过交付期，那么你就可以强制终止讨论，拿出你心中的合理方案，给出理由，要求大家采用。

像C5、C8这类已上升到谩骂或肢体层面的冲突，作为管理者，你应该立即采取强制措施，物理分割冲突双方，避免进一步的恶劣影响。隔离开冲突双方后，再来了解冲突的原因，然后在双方之间居间调停，或者根据公司规章制度，强制性处理，比如罚款、辞退等。

当事情重要但不紧急时，可以努力寻求以合作的方式解决冲突。

回到C1这类冲突，假如当下处于技术预研和选型阶段，离这个阶段结束还有2周，那么你作为管理者，就可以让张三和李四继续竞争，你来澄清项目的目标，设定要考虑的因素，引导大家围绕着项目目标

来分析，这样冲突就可能通过团队成员的相互论证而以合作的方式解决掉。

在你的引导下，张三和李四按照第1章的1.6节提供的思路来分析，发现团队里有人懂JavaScript，有人熟悉MySQL，没有人用过PHP，他们在快速实现MVP的目标下很快达成了一致：用Node.js+Express+MySQL做后端，用AngularJS+ Bootstrap做前端，用Ubuntu Server做服务器，用Nginx实现反向代理。

这样，冲突就以合作的方式得以解决。

回避策略可用于以下情况：

- 冲突的事件微不足道，不重要也不紧急；
- 问题根本没办法解决；
- 解决问题的时机还不成熟，收集信息比立刻决策更重要；
- 冲突双方都在非理性的情绪中；
- 处理这个冲突会可能引发一个更大的冲突。

像C4这类冲突，有可能开发人员和产品经理会采取回避策略，谁也说服不了谁，那就先放着。然后等到快要交付了，必须得解决了，又可能找到开发经理，开发经理又可能采取妥协策略，双方各让一步，折中实现；或者采取强制策略，逼孙八去研究怎么实现。

如果别人给你带来麻烦，但这种麻烦你可以承受，你真的理解别人得到冲突的利益更重要，你愿意维护关系的融洽胜过理性上的对错，那么就可以采取退让策略来解决冲突。

C3中的袁大头就采取退让策略解决了“领导让加班他不想加班”这个冲突，因为他觉得项目进度更重要。

作为管理者，如果团队成员之间发生了冲突，事情本身不是那么重要也不是那么紧急，但双方都不愿意让步，找到你协调解决，这时可以分别了解双方的诉求，看看能否通过沟通，让一方做出让步，最终让冲突以退让的方式解决。

假如你和某个团队成员发生了冲突，也可以考虑引发冲突的事情到底是否重要，你是否有坚持自己要求的必要，如果没有必要或者影响不大，就不要在这类事情上和团队成员争个你死我活，而适当做一些让步，让开发者的要求得以满足，能促进你们之间的关系。

这也是技术管理者应该意识到的一点：不是什么事情都有必须坚持的原则，不是什么时候都必须要赢过下属。

妥协策略可用于下列情况：

- 目标十分重要但过于坚持己见可能会造成更坏的后果；

- 对方做出承诺不再出现类似的问题；

- 时间十分紧迫必须尽快采取一个妥协方案，问题又不是原则性问题；

- 问题很复杂，在要求期限内很难完美解决；

- 事情紧急但不是很重要，或者属于非原则性问题。

C7这类冲突，在软件研发团队中经常会出现。很多软件系统出现Bug，都找不到真正的原因，只好通过贴膏药来避免问题出现。这样并

不能从根本上解决问题，但是考虑到线上缺陷的时限要求，或者版本的交付期限，你只能选择妥协——即接受这种贴膏药的修复方式。

但作为技术领导者，你一定要意识到，这种贴膏药的方式是有后患的——因为问题没有真正解决，程序员添加的类似膏药的代码，很有可能只是堵住了触发Bug的某一个条件，很可能还存在其他触发Bug的条件，在将来的某一刻，它会再次引爆这个Bug。所以，当这个时限紧迫的事件过去之后，你还是要安排Bug所有者查找真正的原因。

作为管理者，冲突处理是必修课，只有恰当地处理冲突，才能维护关系健康，促进团队合作，塑造团队文化。当你再次面对冲突时，一定要先停下来想一想：

- 冲突的原因是什么？
- 自己准备用哪种方式处理冲突？
- 还有更好的处理方式吗？

这样会有助于你用恰当的方式解决冲突，避免给团队造成严重的不良影响。

虽然管理者可以解决各种冲突，有些积极的冲突也可以促进工作，但是有些不必要的冲突是可以提前预防或者通过在早期介入来避免恶化的。比如C8这类冲突，其实管理者在安排任务时，就可以考虑两个人是否能顺畅协作；这类冲突如果不能在一开始预防，也可以在项目进行的过程中，定期了解工作状态，协调不同开发者之间的进度，避免等停造成冲突。

4.9 挑战9：绩效变差

2008年—2014年，我任职于西安信利软件，2008年、2009年我负责做具体的开发工作，每次的绩效评估（半年一次），都是A或者S，奖金系数在2到3之间，2010年我升任研发部门经理，当年绩效考评，结果为C，奖金系数为0.7。

看到这样的结果，我当时很郁闷，因为在我的观念里，管理者应该比执行者有更好的薪酬和绩效！

当时我还因为此事找领导聊了一次，现在想想，当年是如此少不经事——完全没有意识到研发经理和开发者的区别！

我在最开始时，是在用开发者的思维做研发经理的工作，沿袭了做开发时的习惯，把主要精力放在了架构设计和核心模块代码实现上，只是在不得不做管理时才处理一下问题，结果整个部门其实是经常处于无序状态的。我真正应该做的是，把自己当成催化剂，激发部门员工的潜力和积极性，让团队的产出和质量获得提升。

现在回想，我得到那样的绩效结果是合情合理的，因为我的管理工作一塌糊涂啊，我没有很好地完成从自我管理到管理他人的角色转变。另外说实话，我也不具备管理技能，没有经过任何的训练和学习就想做好管理工作，这简直是痴人说梦呢。

我的糟糕经历可以说明一个问题：从技术岗位晋升到管理岗位后，你往往是不能胜任工作的，在上任后的那个绩效评估周期内，你所得到的结果，大概率是比你做普通员工时差的。

面对这样的时刻，你肯定会有情绪，这是自然的反应，可以理解。但在情绪过后，你应该换个角度来看：其实在技术领导岗位上，你是从0开始的，有一个爬坡曲线也是符合逻辑的。

有了这样的经历，你可以更好地理解技术领导者的工作重心是什么，也能促使你有意识地去学习领导和管理技能（关于如何学习和提升，请参考挑战1）。

4.10 挑战10：担心失去技术竞争力

很多技术领导刚刚开始带团队时，往往还停留在过去的角色里，认为技术是唯一的立身之本，担心放弃技术细节后，自己会丧失竞争力，会贬值。

比如会担心万一自己从这个经理岗位离开，就可能既找不到管理岗位的工作，又因为生疏了技术而找不到技术岗位的工作。

所以，技术管理新人会陷入纠结中，一方面想提升整个团队的工作效率而不得不做很多的组织、激励、领导、协调等工作，花费大量精力；另一方面，这些非技术方面的工作会占用他们大部分精力，导致无暇深研技术而产生焦虑感。

其实，此时更重要的是视野。你可能对技术细节了解得少了，但对技术方案选择、技术类别、技术的影响力等可能了解得更多，会形成更为广阔的视野，这足以弥补你在技术深度上的欠缺。而且，其实你之前达到的技术深度仍然存在，甚至会发酵，反过来滋养你的技术视野，因为如果你之前在技术上达到了一定深度，一定在学习上摸索到了适合你的规律，这种学习模式，会帮助你更快地了解更多技术，让你从广度上来丰富自己，这虽然不能保证让你在技术方面更有竞争力，但也会帮助你将技术竞争力维持在某个水平。

最重要的是，除技术外，你在管理岗位上的锻炼，将来一定会带给你更深层次的变化：要么你培育了组织能力、领导能力；要么你认识到自己更适合做什么，对自己的才干和能力边界有更为清晰的认知，而一旦有了这种认知，再做其他事就会得心顺手——因为，你会更容易找到自己喜欢做的事情并带着热忱义无反顾地投入进去。

4.11 挑战11：有效的反馈机制

在挑战9（绩效变差）中，我展示了自己刚做研发经理那一年绩效变得很差的经历，实际上，角色转换期，是可以借助有效的反馈机制来缩短的。

这里说的反馈机制，指的是一组渠道和方法，能够定期对你的行为和你所做的事情进行评估，给你提供评价和建议。

你有了反馈机制，就可以知道自己做得怎么样，哪些地方好、哪些地方坏，找到改进的方向。

在构建反馈机制时，可以从4个维度来考虑：

1）**自我评估。**这里有两方面，一是项目评估，在开始项目时，会界定范围、目标、测量指标、交付时间，当项目结束时，就可以收集项目数据，看看和既定的目标之间有何差距，然后分析差距是怎样造成的，找到原因（可以采用鱼骨图），制定改进策略；二是关键目标分析，可以采用MBO（目标管理）或者OKR（目标和关键成果），比如你把建立代码审查机制作为你的关键目标，那么就可以列入MBO，制定实现策略和结果指标，然后就可以周期性地评估做得怎么样。

2）**来自下属的反馈。**收集下属反馈的方式很多，如一对一谈话（参考挑战13）、周期性会议（周会、月会、季会等）、专用邮箱（匿名）。关键是，你要让大家觉得你真心想要反馈，并且无论他们说什么都是安全的。否则大家什么话都不会说。

3）**来自同级的反馈。**同级可以是关联部门的负责人，比如测试、产品、需求、售前、运维、售后人员等，方式可以是一对一谈

话、邮件等。

4）**来自上级的反馈。**非常重要，一定要主动、持续、周期性地让上级给你反馈，往往他一句话就能让你迷途知返。

在向他人搜集反馈时，一定要事先准备一个话题列表，这样可以提高沟通效率。

搜集到反馈信息后，要进一步分析，分出正面、负面两类。

对于正面反馈，琢磨：

- 好在哪里？
- 做了哪些事情导致了好的结果？
- 在什么样的条件下，可以重复类似的策略，收获类似的结果？

对于负面反馈，琢磨：

- 坏在哪里？
- 做了哪些事情导致了坏的结果？
- 做错了什么选择？这些选择，是在什么情况下做出的？
- 如果重新来过，该怎么做？
- 以后再做事情，这些反馈中，有哪些可以规避？怎样规避？

只有将反馈与经历结合，炼化成经验和教训，才能用来指导后续的行动，才能不断进步。

4.12 挑战12：别人的议论

新晋升的技术领导，往往会因为以前没有做过，而特别在意自己是否做好了，既会担心领导对自己的评价，也会担心下属对自己的看法。这个阶段，风吹草动都会让人浮想联翩。心思较多、比较敏感的人，还可能会因过于忧虑而导致神经紧张。

其实，大风吹倒梧桐树，自有别人论短长。无论你做什么事情，都不可能符合所有人利益，总是有人会议论的，为此而战战兢兢实无必要，还是信奉这句话吧：走自己的路，让别人说去吧。

如果你做不到“但行前路”，可以试试下面3个策略：

1）**观察事情的结果是否到达预期。**如果团队的关键事情都获得了预期的结果，那么说明你做得还不错；如果事情的结果总是达不到目标，一定是因为你的管理工作没有做到位，请用下面两个策略来获得数据，分析原因。

2）**定期和下属进行一对一沟通，获取他们对你的反馈。**这一点很难，但必须要做。关于怎么做，请参考挑战13。

3）**定期和上司沟通，了解他对你的评价，聆听他的建议。**你的上司在管理和人事方面的阅历，很可能要比你强很多，应该定期和他聊一聊，一方面汇报自己的工作，让他知道你的状态，另一方面也可以把你在管理过程中遇到的困惑抛出来，请他点拨一下。

4.13 挑战13：和下属进行一对一沟通

你要求下属加班，下属用沉默对抗……

你观察到胡大侠最近一周迟到了三次，有三个任务延期交付，你认为他的工作效率出了问题，怕他手上的任务出问题影响项目进度，想找他谈谈，可不知道怎么谈……

公司规定每个绩效周期结束时，主管都要和下属面谈，可是你给张三丰的绩效是C（一般），你担心面谈会场面失控……

你想知道下属对你最近三个月的管理行为有什么反馈，却担心他们不愿意开诚布公地向你提出来……

当你成了技术领导者之后，就不得不经常性地和下属进行一对一谈话。可是一对一谈话是相当艰难的事情……所以很多技术管理新人都回避和下属进行一对一沟通，放弃了增进彼此了解、建立信任的机会，最终让团队越来越消极，越来越没有战斗力……

要想做好一对一沟通，首先应该在对话前问自己下面的问题：

- 希望为自己实现什么目标？
- 希望为对方实现什么目标？
- 希望为我们之间的关系实现什么目标？
- 要实现这些目标，自己该怎么做？

上述问题来自《关键对话》，多问问自己，找到谈话的目的和实现目标的策略，准备支持谈话的事实素材，然后你可以参考下面的谈话过程：

• **在开始时阐明目的，为沟通定调。**比如你可以说，“今天我找你是想请你帮助我，让我知道这段时间我在管理上做的事情有什么效果”或者“我今天是想和你谈谈你最近的工作结果，一起讨论一下下一步的工作安排”或者“今天我找你是想请你参加这次的职级评定”……

• **分享事实经过和你的想法。**比如你可以说，“这个月初我们引入了每日站会，我每天都做会议记录，发现有三位小伙伴上一周每天都说自己在改Bug……我很担心这个站会给大家带来负担，因此想……”或者“我查看了项目计划，你的A任务设定的是周二交付，B任务设定的是周三交付，今天周五……我担心你遇到了什么困难，也不想项目因此而延期……”

• **征询对方的观点，鼓励对方做出尝试。**比如你可以说，“我想知道你对这件事的看法”或者“我希望听你坦诚地讲讲发生了什么事情”……在征询对方观点时，多用开放式问题。

注意在谈话过程中要保持尊重，不要评判对方的人品，不要随便打断对方的说话，不要随便评判对方的话。

假如对方有顾忌，暂时不想说，不要强行推进谈话，要先停下来，了解他在顾忌什么，用引导式问题探索对方的需求，比如封闭式的问题“你是在担心……吗”，比如开放式的问题，“看起来你在顾忌一些事情……”。

4.14 挑战14：怕犯错

你刚走上技术管理职位，因为对岗位职责不甚了了（参考挑战3），眼前一片茫然，这个时候就会担心犯错，担心一不小心搞错了什么事导致领导不待见，又因为对上级不了解而很难明了现在的上级的行事风格、是如何要求下属的，自然也担心如果自己的风格和领导不匹配，是否会让领导对自己的错误有过激反应。

还有，你也可能会担心领导对自己评价不好——因为在这个过程中，很多事情做起来没那么得心应手。

你有这种种担心的时候，就会倾向于有把握能做好时才采取行动，就会越发想要用足力气把事情做完美，然后，要么迟迟不能决策，要么劲儿用过头把事情搞错了，最后反倒真的不好了。

其实犯错也是一种成长，没有犯错就很难成长，把错误当作练习和机会，错误就可能变为财富。

要想少犯错、从犯错中成长，请记得以下几点：

- **勇于承担责任**，遇到问题，出了差错，少辩解，多负责，多想想如何解决。

- **时刻牢记公司目标、团队目标、项目目标**，不要总盯着问题，不要陷入到具体问题中跳不出来，要学会回到目标来重新思考你的做法、团队的做法是否正确。

- **因人而异进行管理**，每个人都是不同的，要用适合的方式来管理，比如对喜欢技术挑战的开发者，就给他有难度的任务；对思想单

纯、服从性强的开发者，就多给执行层面的指示；对独立性强、喜欢自由度的开发者，就设定目标让其自行寻路实现……做管理一定要在人身上多花心思，要去研究你的下属，从他们的行为、动作、眼神、语言、思想上去了解和判断，摸清每一个人的习性。

- **重视思想和认知的改变**，结果是由某种行为导致的，行为是由想法触发的，想法是由思想产生的。一个人在思想上不认同你、不认同团队的目标，就很难很好地去执行，自然也很难取得好的结果。所以在制定目标和委派任务之前，一定要统一思想，让大家认同要做的事情。不但如此，还要在日常工作中随时随地因人而异地启发教育下属，影响他的思想和认知。

- **建立反馈机制**，参考挑战11。

- **善用清单**，委派任务、项目验收、发布软件……都可以先列清单，再依据清单检查执行，就可以避免疏漏，少犯错误。

- **复盘**，定期复盘（每天、每周、每月、每季度、每年等），按项目里程碑复牌，突发事件复盘，养成复盘的习惯，这样才能越来越好。关于如何复盘，参考第2章的2.6节。

4.15 挑战15：时间管理

在分答上我收到过一个很有意思的问题：

领导下班总是走得很晚，我不想加班可又担心走得早领导有意见，怎么办？

我在公众号“程序视界”里调查过这个问题，很多人都有这种困惑。不知道你有没有想过：领导下班总是走得很晚是因为什么？

通常，领导下班了老待在办公室，是因为时间管理没有做好。

- 当你成为领导后，时间就会被切分成5部分：
- 老板占用的时间（用来完成老板所要求的工作）。
- 组织占用的时间（平级部门之间相互协调支持）。
- 外界占用的时间（客户、供应商、投资者等占用的时间）。
- 下属占用的时间。
- 自己可支配的时间。

用饼图来表示，如图4-3所示（比例仅供参考）。

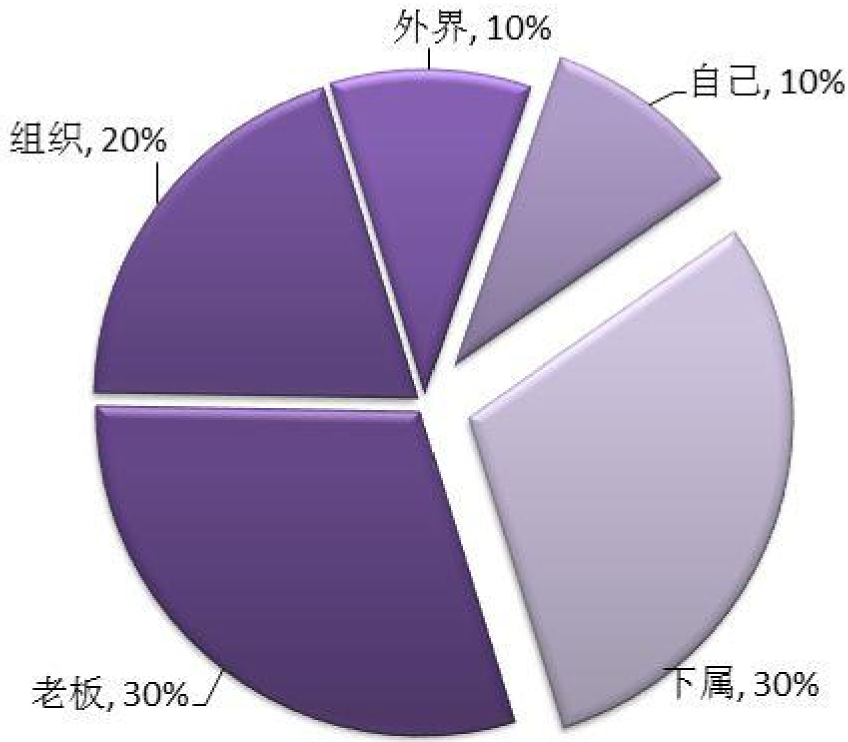


图4-3

一个技术领导下了班总是不走，大多数情况下（排除工作狂）是因为白天自己可支配的时间太少，自己的工作无法完成，不得不推迟到晚上来做。

2009—2011年，我的职位是研发经理，管理20来个人，白天要么在和老板开会，要么在和产品或者市场部门开会，要么和研发部门讨论，要么在帮某个开发者处理问题，即便是坐在工位上了，也要处理大量的邮件，一天下来，忙忙碌碌，总是没时间做自己必须亲力亲为的事情——比如规划各个产品版本的开发计划、制定计划、审核开发者的设计文档、更新项目状态、撰写代码等，而这些必须亲自做的事情，就只能推到下班后、干扰少的时间来做。

在我爱人怀孕的几个月里，我也是这种状态，甚至还要她挺着大肚子给我做饭，想想都惭愧。

实际上这种状况是可以通过合理管理你的时间来改变的。

老板、组织、外界占用的时间，都很难改变，你可以改变的，就是下属占用的时间。管理者如果不能有效管理下属占用的时间，很可能就没有自己可支配的时间，就不能完成自己要做的事情，就会完全被他人支配。所以，**时间管理的重点就是：管理下属占用的时间。**

管理下属占用的时间，重点是交办与授权（参考挑战1和挑战2）。很多管理者一方面不放心把事情交给下属去做，总担心他们做不好；另一方面又觉得下属总是不成长，什么事都得靠自己亲力亲为。其实你不必等到对他们有信心时才让他们做，你只要大胆地、放心地把事情交给他们去做，他们就能想办法完成事情，并且能够在做超越自己当下能力的事情中快速成长起来。

当你把事情委托给下属去做时，就要学会授权。哪些事情他们可以自己做决定（不用来问你），就让他们自己决定，你只需检查结果即可；哪些事情必须经过你的允许才能继续，就说明白，让他们在关键节点来找你讨论，讨论之后的下一步，依然要落实到他们身上去执行。

如果你不懂怎么授权，可以从下面这句话开始，多练习练习：

“X、Y、Z，由你来决定，我完全不需要参与。至于A、B、C，请你来找我确认。如果你在其他方面需要我的意见，当然可以来问我，但除非你认为这样有用，我才会乐意帮助你。”

在做授权练习时，一定要克制住出手帮下属解决本该他解决的问题的冲动——技术出身的你太习惯、太渴望通过帮别人来搞定问题了，你太喜欢那种被需要的感觉了。

当你能够顺畅地把这句话变通应用在各个下属、各种场景中时，你就学会了授权。

推荐两本书，对交办和授权非常有帮助，一本是《交办的技术》，一本是《别让猴子跳回背上》。

能做到交办和授权，下属既可以成长，又可以少占用管理者的时间，管理者就有更多的可支配时间，用来完成自己的事情，比如优化团队目标、配置资源、创造更好的环境、改善工作流程、制定相关政策、琢磨每个员工的特点并制定相应的培养计划等，而且这些才是管理者应该做的工作内容，也是让未来超越现在的关键。

4.16 挑战16：激励他人

一位刚做主管的朋友在微信上给我留言："你们都是怎么做绩效考核的？感觉同事积极性不是很高，安排的任务按说当天都可以搞完，如果没搞完，宁愿去玩儿也不加班，第二天接着做昨天的任务……害得我这边又要加班抽时间搞。"

我回答："其实这和绩效考核关系不大，关键是你的同事得认同你给他的目标，得认为这些任务是他的事情，在一开始分配任务时就要做到这一点；另外就是要让他认你这个人，你委派任务他才会愿意接受。"

他说："我不知道怎么做到这些，我也害怕沟通，不知道沟通什么，正在慢慢尝试……"

我说："下属积极性不高，其实很重要的原因是领导在沟通、目标统合、管理等方面做得不够到位，还有很多可以改进的地方。其中，你是真的关心下属的需求、发展、目标，还是只把下属当作完成任务的机器，有很大差别。只有下属感到你真的关心他、认可他时，他才会积极。"

微信里的三言两语，很难把激励他人这件复杂的事情说明白，因为这里会花比较长的篇幅来探讨这个话题：怎样才能有效激励一个人积极投入工作？

1.传统的经济刺激理论

1976年，经济学家迈克尔·詹森和威廉·麦克林提出"经济刺激是主要激励因子"，之后这一理论被反复引用，也被管理者广泛应用

于企业管理，很快形成了社会共识。它认为经济刺激能有效调节甚至决定人的行为模式，因为人们是“拿多少钱办多少事”的。

过去的实际情况也表明，很多员工不断追求加薪，不断提出加薪请求，不断跳槽谋求更高薪水，这似乎也印证了“经济刺激能够激励员工积极投入工作”的激励理论。

然而如果你在公司内留心观察，就会发现：

• 薪水较低的员工，加薪、发奖金后能够持续一到两个月比以往更努力地工作，两个月之后就会恢复原来的样子。

• 薪水较高的员工，加薪、发奖金后只有两周甚至更短的时间（几天）表现出积极性和努力程度上的波动。

为什么？

2. 大棒

“要使驴子前进，就得在它前面放一根胡萝卜或者用一根棒子在后面赶它。”这就是经典的胡萝卜加大棒理论。

经济刺激属于胡萝卜，前面说过了，这里单说大棒。

大棒政策源于这样一个假设：只有当人们受到生存威胁的时候，才会集中精力、激发思维、提高效率。

所以，很多管理者以为，大棒会给员工带来一种恐惧感，员工会在惩罚的威胁下努力工作。

美国哈佛大学克莱默教授的一项研究表明，很多人喜欢给比较凶恶的和比较严厉的管理者做事情。这似乎也证实了“大棒政策”的有效性。

然而现在社会已经与以往不同了，经济高度发达，物质极大丰富，工作机会到处都是，个体切换工作的成本很低，对大棒已经不敏感了。如果你留心观察就会发现，严厉的惩罚措施只能让员工表面上低头沉默，转身就会抱怨，就会采取看不见的报复策略在水面下积极对抗。

胡萝卜不行，大棒也有问题，怎么办？

3.动因理论

有一个有趣的实验：

把一群孩子分为两组，分开来玩拼图游戏。第一组孩子每拼完一幅拼图，就给他们一美元。另一组就是让他们玩儿拼图，想怎么玩就怎么玩，不给钱。

三个小时后，同时向两个组的孩子宣布活动结束，有趣的现象发生了，第一组的孩子，扔下拼图就走了，而第二组的孩子，则大部分都留下来继续玩。

为什么会发生这样的现象？

答案很简单：第二组的孩子就是觉得玩拼图这件事本身很有趣，就是想玩。

这个实验隐含了动因理论。

以弗雷德里克·赫茨伯格为代表的动因论者认为：人们需要报酬，但物质激励不是真正的“动因”，人们做某件事的动因是：**发自内心地想做。**

动因理论指出了选择工作的两种不同因素：**基础因素和动力因素。**

基础因素即保障因素，包括地位、薪水、安全保障、工作条件等。保障因素不好，会让人不满，但只有好的保障因素，还不足以让你爱上工作。真正让人们满意并爱上工作的，是动力因素（激励因素），它包括：有挑战性、获得认可、责任感、成就、个人成长。

根据动因理论可知：超过一定的临界点，改善保障因素带来的激励效用是递减的。这正好可以解释经济刺激和大棒政策为什么会失效。

4. 工作的隐性价值

薪水是我们找工作时的保障因素，是显性的，那些激励因素（**有挑战性、获得认可、责任感、成就、个人成长**）则是隐性的。保障因素到达临界点后，就不再能有效激励员工，此时动力因素就显得更为重要，它就决定了员工对工作的投入程度。

1）有挑战性

当一个人的工作缺乏挑战，无法让他发挥自己的潜力，他觉得这份工作不能发挥他的才能时，他就很难对这样的工作产生兴趣，他会觉得这样的工作一点儿也不重要，是对自己人生的浪费。一旦产生了这样的想法，他就会对工作失去热情，很难投入进去，就会讨厌这份工作，敷衍了事。

十几岁的孩子不喜欢婴幼儿的玩具，员工的工作必须对其本人有挑战性。

2）获得认可

同事和领导对我们的看法会影响我们对自己的看法。如果每个人都认为自己的工作能够赢得别人的尊重，获得别人的认可，那么就会变得更加快乐，工作也会更加努力。

3）责任感

当一个人对他要做的工作产生责任感后，就会主动投入时间和精力，努力把工作做到最好。想想看一个父亲或母亲是怎么对待他们的孩子的：不需要任何利益驱使，也不需要任何人敦促，他们会自发自愿地尽自己的最大努力来培养孩子。因为生养孩子是他们自己做出的选择，孩子是他们自己的，他们对孩子负有责任。

工作也一样，当我们能够自由选择工作内容、工作方法、工作目标时，我们就会对工作产生更强的自主意识，就会觉得自己能掌控工作的局面，就会产生强烈的责任感，就会更加努力地工作。

有选择，才有责任感。

4）成就

我们想知道自己的努力是有效果的，需要看到自己努力的成果，不断地通过成就获得正向反馈与激励。

当我们能够看到、触碰到、测量、计算出我们的劳动成果时，就会获得更大的满足，就会在一个成就之后追求另一个成就，形成正向的循环。

5）个人成长

一个人想从事具有挑战性的工作、想获得成就、想获得认可，其实都是对自己个人的成长有追求。每个人都希望自己有机会提高自己的能力，把擅长的事情做得很好。

假如一个人所做的工作就是挖坑然后填平，不断循环，那么他很快就会厌倦，因为他从这样的工作中无法获得个人的成长。

每个人都希望能够在工作的同时自己也获得成长，每个人都希望自己变得越来越有价值。

如果一个人能够在工作中体会到上述的隐性价值（哪怕只有一点或两点），那么他就会更加积极地投入工作。

5.管理者如何创造隐性价值

管理者应该努力为人们提供具有隐性价值的工作，同时也要在工作方法和流程上让员工体会到隐性价值。

具体来讲，可以从以下几个方面进行改善。

1）了解员工为什么在这里工作

员工每周要工作40个小时，这40个小时是其生活中可支配时间的一大半，当他选择把这40个小时投资到眼下的公司、工作中去时，他一定期望获得收益。他期望的收益是什么？包含哪些方面？管理者需要搞明白，只有搞明白员工为什么在这里工作（而不是换一家公司），你才可能在工作中改进方法、流程、任务分配模式等来契合员工的需求，只有员工的需求与公司的目标、文化、制度能整合在一起时，双方才能共赢。

2）让每个人都有参与感

管理者可以请大家共同制定团队目标和个人目标。即便是上层领导强制分配下来的目标，管理者也可以和下属一起讨论，让员工参与上层目标分解到团队目标、团队目标分解到个人目标的过程。

参与感可以产生责任感，责任感会产生动力。

如果人们参与了目标的制定，他们就会努力实现这些目标。

如果一个人参与了一项计划的制定，他就会努力实现这个计划。

如果团队的全体成员共同参与了总结以往表现、制定改革计划的会议，那么他们更有可能接受新的工作方案。

当一个人参与到目标、计划、工作方案的制定过程中时，他就会：

• 觉得自己能够在关键事务上发表意见，获得了认可。

• 感觉自己受到了尊重。

• 觉得这些东西是自己的，产生责任感。

• 因为自己掌控了某些局面并产出了东西（目标、计划、方法等）而产生成就感。

所以，只要让员工参与进来，就可以让他感受到至少3种隐性价值，他就会对工作更加投入。

3）改善分配工作的方式

很多管理者日常都在摊派工作，自己分好工作任务丢给下属。下属是被动接受方，没有参与感，所以积极性不高，不能努力投入工作。

还有，我们在分配工作时，往往会有下面的习惯：

• 谁做这种事情最擅长就给谁做

• 谁离自己近就给谁做

• 谁最努力就给谁做

• 谁不抱怨就给谁做

第一个习惯具有多重杀伤力：最擅长做某类事情的人会被限制在这类事情上，没有机会从事更具挑战性的工作，很难获得进一步的个人成长；想做这类事情的其他人员没有机会做，也无法得到锻炼和成长。

第二个习惯往往是因为某个人刚好在管理者身旁经过或者工位离管理者近，管理者顺手就把有些任务交代给他："张三你帮我们预订一下会议室""张三你问一下李四，他的PPT准备好了没有""张三你汇总一下大家的工作进度，下午开会前发邮件给我"……就是这么随意。

至于第三、第四个习惯，此处不展开说了。

总之这些习惯和方法忽略了执行任务的个体，让管理者的任务分配变得不那么有效，甚至低效。

分配工作有三个目标：

1）每个人都有能力完成他所分配的工作任务。

2）每个人的任务都有足够的挑战性。

3）每个人都尽可能地投入工作中。

要达到这些目标，必须让每个人都参与到工作任务的分配过程中。

每个人最了解自己所拥有的能力，最了解哪种工作最能激发自己的工作积极性。如果管理者在**分配任务时能征求每个人的意见，**那么任务就有可能到达最适合做的那个人那里。

为确保需要完成的工作都能得到完成，管理者可以**先把所有任务列出来，然后进行分配**，确保每个任务都有人负责。

上面的两点结合，就会形成**“每个人从任务池中挑选适合自己的、自己想做的任务”**的氛围，而团队的成员之间也会考虑其他人，考虑每个任务都有人负责的原则，不会只挑自己最想做的，还会妥协，接受一些自己不那么想做的。

当然有时管理者需要承受一些压力，比如某项技能还不熟练的员工想挑他目前做起来比较吃力的工作（有其他员工可以熟练的处理这个任务）来锻炼自己，那么这个任务的完成周期就会较长，就可能带来交付上的压力。

不过这是管理者必须承担的，因为管理者的职责有两块：完成工作，培养下属。假如一个管理者只愿意驱动下属完成工作，而不愿意给下属成长的机会，那么他就不是一个合理的管理者，最终也会承受这样做的恶果：团队成员的整体业务能力停滞不前、人心涣散、没有协力。

所以，为了改善工作分配模式，管理者必须要有责任感并能承受压力。而这样做绝对是值得的：一旦实践了这样的任务分配过程，每个员工的意见都得到了尊重，每个员工都有了选择的机会，每个员工都有可能从事有挑战性的工作，每个员工都有可能在自己想要的方向上获得成长，那么整个团队的凝聚力和战斗力就会越来越强。

4.17 挑战17：向上管理

2015年年底，我应聘某上市安防公司西安分公司的研发总监职位，面试时现任总监让我谈谈对向上管理的理解，我只谈到了目标和预期管理，效果很差。

回来一查资料，大感惭愧——我做了6年技术管理工作，居然从未认真了解过向上管理是怎么回事，而向上管理，其实是每一个职场人士都应该学会的技能。

向上管理指为了给公司、给上级及自己取得最好的结果而有意识地配合上级一起工作的过程。

这里有一个问题清单，可以帮助你检视自己在向上管理方面的状态：

- 你的表现符合上司的期待吗？
- 你的上司信任你吗？
- 你了解上司这个人吗？
- 你了解上司的管理风格吗？
- 你能说出上司的优点吗？
- 你了解自己的需求与期待吗？
- 你与上司讨论过自己的职业生涯规划吗？
- 你了解上司的文化背景吗？

• 你能适应与多位上司共事吗？

如果你很难回答这些问题，那么说明你正面临向上管理的挑战，有很长的路要走。

从执行者晋升来的管理者，一定要明白向上管理的重要性，必须有效地与老板配合，高效率、高质量地完成老板要求的工作。你的工作，就是为了让老板的工作更顺利。

通常来讲，你需要做到这些：

• 喜欢你的老板，不要抱怨或者恨他，至少要理解他。否则你就很难积极配合老板完成工作，自然也就很难有好的发展。

• 了解老板的处境，为他排除万难，助他一臂之力，让你的老板工作更有成效。道理很简单，老板升迁，你才能升迁，老板被重视，你才更可能受重视。所以，你要让老板知道，他所重视的，就是你重视的，你会全力协助他达成。

• 把老板当成鲜活的个体，摸透老板偏好的工作习惯，向老板的习惯靠拢。因为你很难为自己量身定做一个老板，你的工作，不是去改造老板，而是在认识到老板在性格与偏好方面与你存在差异的前提下，设法找出彼此磨合的工作方式。请记住彼得•德鲁克的话——下属的工作不是改造老板、再教育老板或让老板依循商学院和管理书籍谈论的模范老板准则。

• 了解老板的强项和弱点，代替老板完成他不擅长的、无法照顾到的或者不愿意做的工作，让老板没有后顾之忧，这样他才可以全心达成他最重要的目标。在老板的弱项上主动承担责任，是最容易让老板知道你的贡献的做法。

• 让老板知道你打算做什么，不打算做什么，正在忙什么，目前处于什么状态。你要主动反馈，这样老板才能掌握你的工作状况，你才不会给老板带来意外，老板才会更信任你，你才可能有机会得到更多的授权。

• 珍惜老板的时间和资源，不要用琐事耗尽老板的时间和精力。你的脑子应该有两张时间表，一张自己的，一张老板的。你要清楚老板什么时候有空听你汇报工作，也要清楚哪些工作只要汇报结果，哪些工作需要提供更多的过程信息。该短就短，当长则长，不要让老板觉得你在浪费他的时间。

• 管理双方的预期和目标。老板每年、每季度、每月、每周，每个项目，都会对你有期望，你要主动请老板告诉你他对你的期望和目标，搞不清楚这点，你要么无法开展工作，要么做不到位，无论如何，老板都会不高兴。如果老板对你的期望和你自己对自己的期望有偏差，一定要及时反馈，争取达成一致。反过来，你要袒露自己，把自己对工作的期望、对职业发展的想法、对老板的期望，都告诉老板，这样他才知道怎么帮助你实现你的目标。

4.18 挑战18：提升领导力和管理能力

《成为技术领导者》这么定义领导的职责：

领导的职责就是创造这样一个环境，每个人都能在其中发挥出更多的能力。

如果你能理解这一点，对培育你的领导力会有相当大的助益。以此为目的，技术领导应该是一个公仆的角色，为团队成员服务，有人需要资源就给协调资源，有人不明白目标就帮助他明确目标并制定其个人目标，不同的模块间接口无法确认就组织相关人员讨论，张三任务完成得好就明确肯定，李四对自己所从事的技术方向感到迷惘就协助他找到归属和发展方向，有人忽然情绪低落效率低下就及时发现背后的原因并在必要时提供支持……大家和你一起同甘共苦完成了一件事，并且都看到并认可你的努力，你就具有领导力了……

总之，你努力创造一个让大家各尽其职各展所长的环境，让这个组织运转正常，让目标得以实现，那么你的领导力就形成了。

但这样的说法实操性不强，所以，我从第3章和前面17种挑战中摘选出来10点以供参考：

- 多和员工沟通交流，了解他的工作，也了解他的生活，坐在角落里盯着电脑屏幕的研发经理，很难和员工建立融洽的关系。

- 与每一个下属同在，真正地关注、关心、倾听对方，让他感受到你是关心他的，你能和他聊他的车子、房贷、喜欢的篮球明星，能理解他陪产的劳累、带娃的辛苦，也能理解他对开发者年龄的担忧前途的迷惘。

• 了解什么能够激励某个人，用他喜欢的方式激励他。比如张三热衷于技术，那么给他充足的时间让他解决一个复杂的技术问题就是激励；比如李四喜欢被大家关注，那么让他做技术分享、演示产品、在会议上介绍方案，就是对他最好的激励。

• 充满自信地与人沟通。

• 与项目同在，和大家一起做事，信任和关系是共同经历出来的，而不是因为职权产生的。

• 目标感强。

• 勇于负责。

• 能承受压力，且不给下属不必要的压力。

• 直面冲突，解决冲突，而不是回避、推卸责任。

• 足够了解公司，理解并接纳公司的愿景，知道公司的价值观，知道什么对公司最重要，熟悉公司的产品，了解行业信息，熟悉公司所在市场情况……你了解得越多，就越能够创建或解释愿景，把所在部门的工作和整个公司的愿景联系起来，这样部门的工作就有意义，每个人也才会觉得他所做之事有意义。

如果你能做到这些，就具备相当强的领导力和管理能力了。

第5章 跳槽8问

对开发者来讲，没有退休，只有跳槽和被辞退。可能你一两年就要寻找新的机会，那么，什么样的跳槽对你的未来发展是好的？你跳槽时都考虑哪些因素（公司大小、城市大小、行业）？你是怎么选的？你的选择是明智的吗？我们来聊聊这些。

要想越跳越好，有两个要点：

- 明确职业目标或者个人未来愿景，指导跳槽。

- 如果目标不确定，就要周期性（每次跳槽时）地回顾，慢慢发现自己的目标。

5.1 为什么要跳槽

作为开发者，你跳槽可能有很多原因：

- 领导差劲，微管理、不尊重人、厚此薄彼
- 技术氛围太差
- 团队里没有牛人
- 产品没前途
- 技术太Low
- 薪水太低
- 薪水倒挂
- 福利太差
- 不被尊重
- 不被认可
- 被边缘化
- 没有发展空间
- 团队死气沉沉，很压抑
- 无意义的加班太多

- 失去激情
- 工作没有挑战
- 过于清闲
- 无法成长
- 行业衰落
- 公司平台很差
- 编程让自己越来越内敛，不能接受
- 年龄大了，干不动了

……

回顾一下工作中让你感到不能接受的事情，一一列下来，形成一个“不能接受”清单，类似下面这样：

- 领导当众批评我
- 领导总是要求我加班
- 张三刚来却比我薪水高
- 公司取消在家办公

……

你的不能接受清单可能很长，你可以对清单中的每一项问三个问题：

• 我不能接受它是因为什么？

• 我愿意因为它离职吗？

• 采取哪些措施，可能会消除这一项？

把你不愿意因为它离职的、可以通过做点什么事情来消除的事情，都从不能接受清单上划掉。重复做这一步，直到你的清单短到只有三项，那就可能是你离职的直接原因。

你如果因为无法忍受某些事情而想离职，那么一定是这些事情触动了你内心的某个原则或你认为很重要的东西，你认为很重要的这个东西，就是你的需求，也就是你的职业价值观。我们需要把它找出来，这样就把离职这件事彻底想明白了。

下面的句式可以帮助你分析直接原因背后的职业价值观：

“某某事情，让我想离职，因为什么”。

对你不能接受的清单上的三项，逐一套用这个句子，细细体会、感受，直到你觉得这个句子不用再修改，就可以把它转移到你的离职原因清单上。

最终你做出来的清单类似下面这样：

• 领导经常当众批评我，让我想离职，因为我感觉不被尊重。

• 团队技术氛围很差，让我想离职，因为我崇尚技术，想做技术咖。

……

当你做出这样的清单后，你就能弄明白你为什么跳槽、你在追求什么，它们就可以作为你选择新工作的标准，你就不容易跳到同一个坑里。

5.2 什么时候跳槽好

“‘金三银四’跳槽会好一些吗？”

“六七月份是不是不好找工作？”

“年底想换工作，是不是很不明智啊？”

类似的问题，我在分答上回答过很多次，我也曾经思考过跳槽和时间到底有什么关系，现在我倾向于认为，它们是内在和外在的关系。

2007年12月，我离开了东大集成电路，是因为和爱人商量好要回西安，她已经先回去，我自然不能久留，越快离开南京越好。

2014年9月，我离开了工作7年的信利软件，是因为我负责的智能机顶盒产品线停掉了，我觉得留在公司已经没有意义了。

2014年11月，我辞去一家公司的项目总监职位，是因为我体会到纯粹的管理工作只会给我带来虚无感，我更喜欢技术带来的挑战和成就感。

2017年7月，我离开全时，成为自由职业者，是因为我再也无法承受组织的束缚，对自由的渴望强烈到了可以接受自由职业的波动性和不确定性。

从这些跳槽经历中可以看出，我们是先有了跳槽的想法，才有了时间选择的问题；而不是因为时间到了，才准备跳槽。从这一点上看，**跳槽的最好时机就是：你觉得真的该离开了，一天也不愿意再待下去了。**

那么怎么判断自己真的该离开了呢？有四种方法（从简单到复杂）：

- 消耗感
- 周末探视法
- 盖洛普的Q12（后面有解释）
- 需求供给分析法

1.消耗感

你留在这个地方，一直在用已经熟练的技能、方法被动地完成事情，不再有成长，你处在营养流失阶段，一点点被消耗。

如果你经常有这种感觉，那么可能这方池塘已经不适合你这条大鱼生长了，你需要更好的水域。

2.周末探视法

在周日的晚上，想到周一要去工作，你的心情是：沮丧、拒绝、低落、痛苦、担忧，还是期待、高兴、兴奋，还是无所谓？

这样做：

- 记录你的感受。
- 探寻是什么（人/事/物/环境/流程等）让你产生了这种感受。
- 改变哪些因素，能够导致你的感受改变。

如果你做了这样的测试，发现自己非常痛苦，压力很大，很排斥来到公司，思来想去也没有哪个因素的改变会让你的感受好一些，那

么可能就到了该离开的时候了。

3.盖洛普的Q12

盖洛普的Q12可以用来测试你对当前工作环境的满意度。12个问题如下：

1）我知道公司对我的工作要求吗？

2）我有做好我的工作所需要的材料和设备吗？

3）在工作中，我每天都有机会做我最擅长做的事吗？

4）在过去的7天里，我因工作出色而受到表扬过吗？

5）我觉得我的主管或同事关心我的个人情况吗？

6）工作单位有人鼓励我的发展吗？

7）在工作中，我觉得我的意见受到重视吗？

8）公司的使命目标使我觉得我的工作重要吗？

9）我的同事们致力于高质量的工作吗？

10）我在工作单位有一个最要好的朋友吗？

11）在过去的6个月内，工作单位有人和我谈及我的进步吗？

12）在过去一年里，我在工作中有机会学习和成长吗？

这真是要命的12个问题，既简单又精确！

建议你每年自测两次，从结果上就可以判断出自己是否还需要留下来继续工作。

4.需求供给分析法

一份工作有很多特质（参考了《生涯咨询与辅导》、《精进》，还有我自己的一些补充）。

1）冒险：工作充满挑战，需要冒险。

2）权威：在工作上运用自己的职位控制别人。

3）竞争：在工作中必须经常与人竞争。

4）创造性与自我表达：在工作中经常能运用想象力，做自己想做的事，说自己想说的话。

5）弹性时间：可以自行决定工作的时间。

6）助人：能够对别人的困难提供直接的帮助。

7）收入：工作能够赚到大钱。

8）独立：自己有充分的自主权，决定要做什么和怎样做。

9）影响他人：工作上能影响他人的意见或决定。

10）智性刺激：工作本身需要相当程度的思考与推理。

11）领导：在工作中能够指导、管理、监督他人。

12）户外工作：工作的地点在户外。

13）说服：工作的性质是说服他人行事。

14）劳动：工作需要用到许多体力劳动。

15）声望：工作能使自己在别人面前、邻里之中有地位、有尊严。

16）公共关注：工作能使自己很快得到别人的注意。

17）公共接触：工作需要经常与公众接触。

18）认可：工作有利于自己变成公众人物。

19）研究：在工作上能发现新的东西然后应用它。

20）例行性：工作有着固定的流程，不必经常改变。

21）季节性：只在每一年的固定时间段才工作。

22）旅行：工作需要经常旅行。

23）变异性：工作的职责经常更改。

24）照顾小孩：工作的对象是孩童。

25）手部操作：在工作中大部分时候需要用到手部的动作。

26）机械操作：在工作中大部分时候需要运用到机械等设备的操作。

27）数字运算：在工作中大部分时候需要运用到统计学或数学。

28）培训指导：有优秀的导师或主管进行一对一指导或可参加有体系的培训。

29）工作强度：工作需要经常性地加班，工作节奏快。

30）团队氛围：和谐、富有生产力的团队氛围。

31）考评制度：有公正、透明、合理的绩效考评制度。

32）晋升空间：公司为员工提供合理的职业上升通道。

33）工作环境：公司为员工提供舒适、环保、健康的工作环境。

34）食物：员工在上班期间能获得营养、健康、美味的食物。

35）艺术性：工作内容与艺术审美相关，以顺带获得审美愉悦。

36）通勤时间：工作地点与居住地之间的通勤时间合理，最好不超过1小时。

37）形象：工作内容使得自己能保持比较好的公众形象和气质。

38）人际关系：自己的性格能够与公司的人际关系氛围相匹配。

39）直接领导：上司行事作风与自己匹配，有较好的领导和管理能力。

40）兴趣相关性：工作内容与自己的兴趣有一定相关性，最好不要是自己排斥的。

41）发展专长：工作可以中可以不断运用到自己的专长，或者自己期望培养的专长。

42）尊重：工作中能够获得同事们的尊重。

43）目标相关性：我将来的目标是从事××，眼前这份工作能否给我所需要的经验。

你选择一种职业，往往是因为你看重什么，也就是你的行为价值观决定了你的选择。上面列出的工作特质，就是常见的“职业价值

观”。还记得“你为什么跳槽”那一节的方法吗？它最终导出的3个离职原因，指向的也是职业价值观。

请仔细斟酌你看重什么。然后，按照下面的步骤来分析自己是否还应该在当下的公司待下去。

【1】

我进入公司时看重的工作特质是哪些？

1.

2.

3.

【2】

我现在看重哪些工作特质？

1.

2.

3.

对比你刚进公司看重的特质和你现在看重的特质，可以感知到自己在工作认知上的变化。这里可以关注自己为什么发生了这些变化，也许会有助于你以后的职业发展。

我们在完成下面第3步时，请以这一步（第2步）为准。

【3】

公司可以满足我看重的特质吗？（针对上一步骤列出的特质，公司还能满足的，打√）

□1.

□2.

□3.

【4】

以你看重的特质为纲，回顾、分析公司的现状，看是否存在某些可以调整的环节，能让你看重的需求得到满足。记录这些环节（以供后面实践时参考）：

1.

2.

3.

【结论】

如果最看重的工作特质，在当前公司已不存在，而且没有调整的可能性，那就到了离职的时候了。反之，则可以继续留在公司。

上述方法的关键点就是：**决定哪些要素是自己最看重的**。这源自于一种认知：世上并不存在完美到让你时刻都满意的工作，觉知自己的关键需要，接纳次要需求无法满足的不完美。

5.3 什么是好的跳槽和坏的跳槽

职业目标，简单地说，就是你要在什么领域、做什么事情、取得什么样的成绩、成为一个什么样的人。

职业目标应当与你的人生目标和规划关联起来，这样你的工作才能帮助你实现人生规划。

按时间可以把职业目标分为短期（1~2年）、中期（3~5年）、长期（5~10年）。现在社会变化很快，不少职业正在迅速消亡，大量新的职业次第涌现，我们很难看到10年、20年之后自己具体在做什么职业，所以，能往后看3年，看到中期目标就不错了。

想想3年后自己在做什么：Web前端开发、电商后台架构师、研发经理还是测试、产品经理？想明白这些，它可以指引你有计划、有目标地更换工作。

需要注意的是，你应该每年都审视、更新你的中期规划。即便现在没有换工作的打算，也要考虑——“3年之后我在做什么”，这样你才可能避开随波逐流的职场大坑，把你自己的成长目标融合在工作目标中，积极主动地去工作。

与职业目标相关的跳槽是好的跳槽，否则，是坏的跳槽。

如图5-1所示，有5次跳槽，公司A、B、E都和职业目标没什么关系，那么就是坏的跳槽。

如图5-2所示的这位朋友，每次跳槽，都沿着职业目标进行，就是好的跳槽。

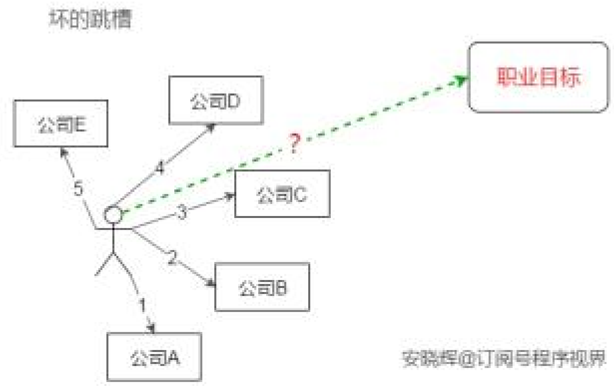

坏的跳槽
公司E
公司D
职业目标
?
4
5
公司C
3
2
公司B
1
公司A
安晓辉@订阅号程序视界

图5-1

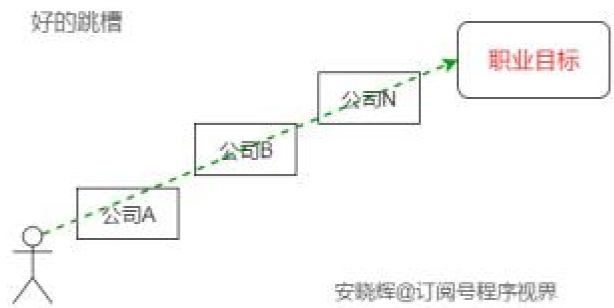

好的跳槽
职业目标
公司N
公司B
公司A
安晓辉@订阅号程序视界

图5-2

5.4 跳槽还是卧槽

跳槽与卧槽哪个好？

关键看哪个符合你的目标。

很多腾讯、阿里、华为卧槽的员工，都财务自由了；也有很多跳槽的人芝麻开花节节高。当然也有一些跳来跳去，最后发现自己比卧槽的同学忽然差了一大截子。

我有两个同学，一个在电力系统的一个下属公司工作，从未跳槽，一个在私企跳来跳去。到2015年时，没跳槽的那位是处长，有三套房子，两个孩子。另一个同学，则干不下去了，找那位处长同学给安排了工作。

再举个例子，2017年3月，21世纪商业评论发布了一篇文章，题目是“一位小米前员工的财务告白：如何处理期权让我纠结”（http://tech.qq.com/a/20170319/ 005866.htm），该文章一时引起热议，新浪、腾讯、中国通信网、爱范儿等媒体纷纷转载。我比较关注这篇文章里受访者对自己职业生涯的反思。

下面是他的跳槽经历：

2008年毕业于北京的一家985大学，进入某大型国企。

2010年跳到BAT中的一家做开发。

2012年跳到亚马逊。

2014年离开亚马逊加入小米。

2016年年底，离开小米到一个家跨国公司工作。

“从外企跳槽到国内公司，其实应该关注的地方有两个，除期权外，要考虑个人的成长性。”

“离开小米时，我不免考虑自己这两年究竟获得了什么——在那两年间亚马逊的股票涨了4倍，虽然当时的股票数量不多，但如果再坚持两年，把剩下的90%的股票拿到，如今可能也够在北京买房付个首付——虽然我是不买房主义者，主张投资股票，这样变现快。

“再回头看，工作8年，两年一跳。职业生涯感觉整体往上走，是不是每次都在正确的时间做了正确的选择？不一定。

“如果2008年不去那家看上去很美好的国企，而是到一家互联网巨头，那职业生涯的起点真的就很高了；从2012年到2016年，如果只在亚马逊或者一开始就选择小米，理论上的收益都比现在高。有的人以不断跳槽获得了最优结果，但并不是每个人都有赶上时代与公司风口的洞察力，或者是运气。”

这位受访者的经历和感慨，说明了跳槽的一个关键属性：不确定性。无论你权衡多少因素，都无法保证这次跳槽一定比上一次好。

所以跳槽时一定要优先考虑某个机会是否有助于实现自己的中长期目标，这样才能大概率地做出有价值的选择，避免越跳越糟，越回首越后悔。

来看一下联想高级副总裁乔建的经历。

根据公开消息，乔健1990年毕业后即以文秘身份加入联想，在购买员工住房时恰巧与杨元庆成了邻居。1995年调入杨元庆手下的销售部门，后被杨元庆委任全面管理市场部门。做了8年市场推广后，2002

年乔健转岗至人力资源管理，并在人力资源方向又工作了10年，2012年升至联想人力资源高级副总裁。

22年，从文秘到高级副总裁。

再看一下曾宪杰的工作履历：

2002年毕业于浙大计算机系，进入上海时佑信息系统有限公司，做系统集成。

2003年9月1日，进入上海先锋商泰电子技术有限公司。

2004年10月离职，作为技术合伙人，参与重庆快点科技有限公司的创业。2007年4月离开。

2007年6月加入淘宝网平台架构团队；2010年1月任中间件团队负责人；2013年5月，成为淘宝技术部技术总监。2014年10月离开淘宝。

2015年3月，加入蘑菇街，2015年3月至2016年6月任蘑菇街技术副总裁；2016年蘑菇街和美丽说合并，任美丽联合集团技术副总裁。

曾宪杰的职业生涯中有两个节点非常重要：

1）加入淘宝。这使得他赶上了电子商务的黄金发展期。

2）在淘宝卧槽7年。这是其职业发展的关键阶段，凭借技术做出成绩，获得声誉，转型管理，进一步积累了声誉和人脉等，并升任淘宝技术部总监。这个时期在技术、声誉和人脉上的积累，为他后来的发展奠定了坚实的基础。

由曾宪杰的经历可以看出，**持续性积累非常重要**。不管是跳槽还是卧槽，我们都要以自己能否在某个方向建立积累为原则。符合你职

业目标的跳槽，能持续为你在某个方向带来积累，能让你的职业生涯更上一个台阶，就是好的选择。

5.5 要不要追薪式跳槽

刚毕业的很多朋友，喜欢哪里薪水高就往哪里去。但实际上，短期的高薪不见得长期有利于你的职业发展。这一点在上一节中讨论过。

最理想的跳槽是：既符合职业发展方向，薪水又高。

如果仅仅是薪水高，背离自己的目标，可能不一定好。举个例子，华为和中兴的海外技术支持，尤其是非洲和中东地区的一些国家，薪水加上每日补贴，整体收入非常可观，可是你愿意去吗？干上三五年后，你准备做什么？

即便你的目的就是获得高薪水，那也不大可能一直通过追薪式跳槽来获得（初期几年可能如此）。如果把开发者作为资源和商品来看，其价格一定是符合市场规律的，最终会回归到他能创造的价值上去。所以开发者的高收益，最终应该是通过个人的成长，通过你能解决层级越来越高、难度越来越大的问题而间接获得的。因为这时你创造了更多价值，所以能拿到更多回报。

我们看一个Offer，不仅要关注薪水数字，更要关注：**它能提供给我们什么发展空间和成长机会。**

只有薪水没有发展的机会，是以牺牲你未来发展为代价的。一时之爽换长期之伤，长远来看是不值得的。

在有发展的前提下考虑薪水，这是我们跳槽时要遵循的原则。

5.6 选大公司还是小公司

国家统计局2011年发布过大中小微型企业划分办法，软件和信息技术服务业的分法如下：

指标名称	计量单位	大　型	中　型	小　型	微　型
从业人员(X)	人	X≥300	300>X≥100	100>X≥10	10>X
营业收入(Y)	万元	Y≥10000	10000>Y≥1000	1000>Y≥50	50>Y

如果这么划分的话，像BAT、京东、网易、360、YY、美丽联合、爱奇艺等就都属于大公司了。但是实际上很多企业我们很难查到它的营收是多少，在找工作时就比较难确定它到底是大公司、中型公司还是小微公司。

所以我们在这里可以换一种简单的分法，从公司人数和主营业务来分。公司人数还是参考上面国家的标准。主营业务状况，可以简单做如下划分：

- 大型公司，主营业务稳定，市占率行业TOP 20。
- 中型公司，有稳定的业务方向，高速发展。
- 小微公司，业务尚处于探索期，不稳定。

接下来分别来说说这几类公司的优缺点。

1. 大公司

大公司优点：

- 平台很好，可以为你的个人品牌背书

• 待遇福利好

• 牛人比较多，可以近距离接触交流

• 稳定

• 更大、更全面的行业格局和视野

• 专业的工作方法，比如开发模型与工程方法

• 能看到多种业务，行业信息储备好

• 复杂而专业的管理流程

• （对新人来讲）学习成长空间大（不像创业公司那样入职就要满血死拼）

• 丰富的资源和人脉

缺点：

• 组织体系复杂，惯性大，跨部门协调沟通成本高

漫长而复杂的流程（因为管理流程复杂而专业，真是双刃·剑）

• 整体效率相对低下

• 非大咖的个人没有话语权，出头难

• 分工过细，非大咖大概率沦为螺丝钉

• 个人技术发展得不全面

• 办公室政治（人多就成江湖）

2. 中型公司

中型公司的优点：

• 业务方向相对稳定，高速发展

• 技术可以做深，因为有很多改造和探索的诉求

• 个人在技术上的成长空间比较大，实战的机会很多

• 技术价值可以体现出来

• 发展空间相对较大，个人相对容易出头

缺点：

• 业务相对（大公司）单一

• 后来加入公司的开发者，如果水平一般，已很难分享公司的前期成长

• 二次创业的风险大（如果和大公司在同一个市场中，就必须寻找差异化方向二次创业）

3. 小微公司

小微公司的优点：

• 发展空间很大（在公司能壮大的前提下）

• 财务回报可能很大（也许过几年就上市了，你也就财务自由了）

• 能得到全面的实践和锻炼（因为你能做的事情很多）

• 管理简单，决策流程短，效率高

• 更大的个人成就感（你做的事情能很快看到成效）

• 出头容易（池塘小）

• 协调沟通容易

• 能经历一个产品的创建和成功（可能还有失败）

• 能接触到创始人（大多数都是精英），可以近距离了解学习

缺点：

• （除了类似声网这类技术方向外的创业公司）技术上很难做深

• 变化多，可能今天做这个明天做那个，因为业务还不稳定

• 风险比较大，100家中可能只有1家能存活下来

• 工作负担重，压力大

• 混乱（缺乏管理）

• 搞定问题靠自己，成长也靠自己，不适合不能自主学习探索的开发者

4. 怎么选择

其实直接比较公司优缺点是无法确定该选择哪家的。

做选择的关键在于：**你自己想要什么工作特质**。这一点我们在5.2节已做过说明。

假如你已明确了你想要的3个工作特质，那么就可以在多个Offer中选出符合你的需求的那一个。

假如你很难确认自己想要什么，但又必须做出选择，考虑到“从大公司进入小公司容易，从小公司进入大公司比较难”这种现状，那就选择大公司，去享受它的各种好处。这样当你发现了自己的职业目标或者厌倦了大公司，需要离开时，这段经历也可以为你的履历增色。

5.7 去大城市还是小城市

城市大小对个人职业选择与发展有较大影响，可以从下面四点来看：

- 城市与产业结构
- 城市大小与公共资源
- 城市与生活成本
- 城市节奏与个人性格

1. 城市与产业结构

地域和产业有密切关系。比如沈阳，机械制造业发达；珠三角，出口加工业发达；东莞，有大量的电子加工、玩具制造企业；深圳，硬件设计和制造发达；上海，软件、金融、教育、公共服务等产业都很发达；北京，软件、教育、医疗等产业优势远超其他城市；榆林，煤炭、石油、天然气等产业有独特优势；大理、丽江，旅游业、服务业发达；……

所以，当我们选择一个城市时，要更多地考虑这个城市的产业结构与自己的专业技能是否匹配，自己在这个城市是否有用武之地。假如你的目标职业是硬件设计工程师，那么可能深圳就是最好的选择。

大城市产业多元化、机会多，这是吸引外来人才的巨大优势，而且这种吸引力具有螺旋增强的趋势：人才越多，相关产业越发达，机会越多，对其他人才的吸引就越强烈。

2. 城市大小与公共资源

很多时候我们选择一个城市，还因为这个城市的公共资源。

比如医疗资源，北京有83家三甲医院，协和医院、友谊医院、301医院等医院的医疗水平全国拔尖。

比如教育资源，全国共有112所“211”学校，北京有24所居首位，江苏11所和上海9所分别排在第二、第三位。全国39所“985”高校，分布在18个省市，排在前三位的是，北京8所，上海4所，陕西3所。

再比如体育场馆、赛事资源，也有很强的集中性，北京奥运会，那是全国之力搞起来的。

再比如创业资源，北京、上海、深圳的双创氛围很热烈，各种扶持政策、配套设施都比西安、长沙这些地方好很多。

3.城市与生活成本

城市越发达，人均收入越高。我的一个小伙伴做Java后台开发，2015年年底，从西安跑到北京，薪水直接翻倍了。然而城市越发达，生活成本也越高。

比如在深圳，你做得不错，可能一年税后收入超过50万元，但要买个学位房要1000万元到1500万元，压力很大。房子的压力，已经成为各大城市外来从业人员的首要压力，北京、上海、杭州、广州，个个如此。

你在北京工作，一开始可能租房子，但房租太贵，可能地下室都舍不得租，往往与人合租或选择偏远地带。想买房子，几环就不用说了，好几万元一平方米，买不起，最终可能选择在通州、燕郊、昌平等地段买，而你的工作地点呢，可能在海淀区中关村，这样一来，通勤成本就很高，一天在上班路上花掉的时间可能在4个小时左右。如果

你搞软件开发，遇到加班较严重的公司，晚上10:30后才离开单位，那么回到住的地方，就后半夜了……

我在西安，午饭一碗面或一份两荤一素的米饭套餐，12元左右搞定，我到上海培训时，大概在20元。大城市相比小城市，吃穿用度都要贵一些，这些都是成本。

还有，你在上海谈个对象，成本比安康高太多了。

4.城市节奏与个人性格

作为独一无二的个体，你其实是有自己的振动频率的。有的人性子慢，喜欢田园牧歌式的慢生活，若放到上海、深圳这些地方，就不适应。有的人性子快，走路都比别人快三拍，在成都、昆明、烟台这些地方，看着人家摆龙门阵、提笼架鸟，会有毋宁死的感觉。

每个人的兴趣、爱好都不同。有的人喜欢文化底蕴厚的地方，在西安、南京、成都等地方就如鱼得水，到深圳、珠海可能就不太适应。有的人深具佛性，偏爱藏传佛教文化，那么可能拉萨是比较适合他的地方。

5.怎么选择

按照“职业=行业职能”的定义，选择职业时，行业和职能是两个基础维度，其中行业受城市影响最大，比如你想做软件开发工程师，在漠河恐怕就很难做；而你想进行硬件创业，深圳可能是最合适的地方；如果你想做职业规划师，那么北上广深这些现代化程度高的城市机会更多（因为人面临的选择多压力大寻找自己定位的需求更大）……

大城市相关产业发达、机会众多、发展空间大，这就是它对职业人士的吸引力。

我们选择一种职业，除了需要考虑行业，还可能会考虑将来的生活，比如结婚、买房子、生孩子、教育、养老、医疗。在大城市虽然机会多，但吃穿用度、教育、医疗、养老等成本也高，可能你在北京月入3万元的生活感受还赶不上西安月入1万元。

所以，怎么选择？最关键的，还是问问自己，你想要什么样的工作，想要什么样的生活，想要成为什么样的人。只有通过这样的自我探索，明确了你的职业目标，才能做出有长远价值的选择。

5.8 自己的选择是明智的吗

我们会在多个机会前纠结，难以选择，往往是因为想要这个选择能有一个确定的、好的、100%有保障的结果。然而，遗憾的是，所有的选择，都可能有坏处。就像我们在5.4节“跳槽还是卧槽”中所举的那个小米前员工的例子一样，你选了当下看着好的，可能将来不好，你离开了当下看着不好的，结果你走后它却可能空前大发展。我们永远也无法给自己的选择买一份全险。我们能做的就是：接纳不确定性，尽量明智地去选择。

那么怎么判断自己的选择是否明智呢？

有选择前和选择后两种方法：

1）选择之前，可以判断这个机会是否符合你的职业目标。符合目标，就明智；不符合，就很可能不明智。

2）选择之后，你可以观察自己的感受。假如你常常后悔，觉得备受煎熬，没有意义，没有动力，无法投入地工作，那么可能你的选择就是不明智的。反之，觉得有意义，有期待感，工作很快乐，被认可、被尊重，觉得自己因为做了这个工作特别有价值感，那么选择就是对的。

第6章 简历优化指南

在找工作时，有人投递上百份简历才收到一两个要约电话，有人一投一个准。其中的差异在哪里？怎样写出让人眼前一亮的简历？怎样提高简历通过率？招聘信息里都隐藏了什么信息？

这是你我都关心的话题，通过本章的学习，你能掌握定制简历的方法，结合你的具体情况，打造你的通关文牒。

6.1 简历优化模型

简历被HR打开的那一刻，她的行为方式是扫描你的简历，用30秒以内的时间来决定是否通过这份简历。她可能只扫描这些信息：

- 目标职位；
- 从研发部门那里拿到的技术关键词，比如Java、Hadoop、FFmpeg、微服务等；
- 学历、学校；
- 工作年限；
- 工作过的公司（名字、规模），有知名公司会加分。

当你的简历通过HR筛选以后，会抵达程序员或者技术经理那里，他们会阅读你的简历，核对知识、技能和项目经历是否匹配需求，在3分钟之内决定是否通过这份简历。

这就是你的简历经历的两道关卡，其中关键不是你技术能力多强或者工作经历多牛，而是匹配度，如图6-1所示。

所以我们的简历优化，一定是针对这两个过程来下功夫，想办法提高匹配性、突出亮点。

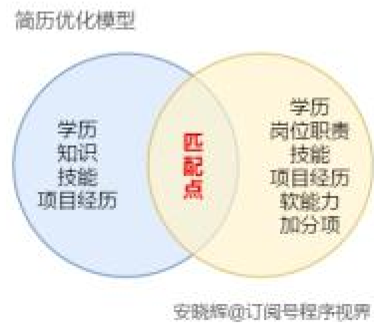


图6-1

围绕着这个核心，我们的简历优化之旅分为下面几步：

1）明确你想要的工作特征

2）盘点你的价值和亮点

3）寻找机会

4）招聘信息分析

5）简历优化

6.2 明确你想要的工作特征

当我们要更换工作时，心中都有期待。你对工作的期待，就是你希望这份工作具备什么特征。

比如说高薪水、大平台、离家近、福利好、技术氛围浓厚、团队里牛人多……

只有你明确了对工作的期待，找到了你想要的工作必须具备的特征后，你才能有选择地、有效地寻找工作。

可以从两方面找出你想要的工作特征：

1）你想要什么样的工作

2）你对当前工作有什么不满

1.你想要什么样的工作

你选择一份工作，一种职业，往往是因为你看重这份工作的某些特质。在第5章的5.2节，我们列出了工作的43种常见特质，你可以从中挑选你个人看重的几项特质，一旦挑出来，它们就可以指引你选择适合的工作。

我个人看重的特征：

- 发展专长

- 薪水

- 创造性与自我表达

• 智性刺激

• 通勤时间

注意，这里在选择工作特征时，先不必纠结优先级顺序和数量，觉得比较想要的，就先列出来。

2.你对当前工作有什么不满

其实有时你从正面去想，不一定想得出来你到底想要什么样的工作，或者你会列出很多特征，难以取舍。这个时候，不妨从反面看看，即分析一下你对当前工作有什么不满，把你的不满列出来，这些不满的反面，可能就是你想要的工作特征。把它们列出来，结果可能如图6-3所示的这样。

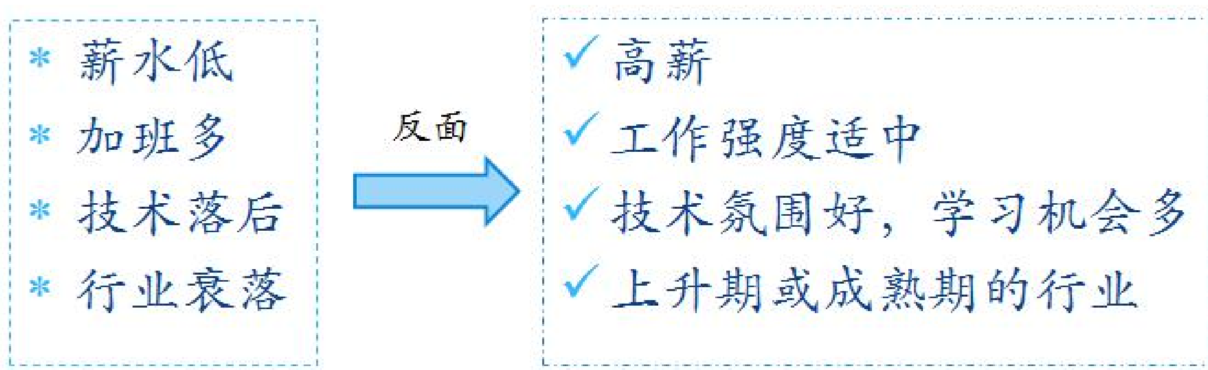


图6-3

这也是第5章5.1节“为什么要跳槽”中，分析跳槽原因时介绍的方法。

3.理想工作的特征清单

从你想要的和你不满的两个角度出发，我们得到了两份工作特征清单，现在把它们合并到一起，从中挑出3~5项来，按优先级排序，它们就是你心目中理想工作所应该具备的特征。

以我个人为例，理想工作的特征清单如下：

1）通勤时间短

2）发展专长

3）薪水

4）创造性与自我表达

5）智性刺激

6.3 盘点你的价值和亮点

找到你的价值，对自己有个客观的认识，这对你撰写自己的简历非常有帮助，也是找到理想工作的关键所在。

个人的商业价值体现在5个方面：

1）知识

2）技能

3）经历

4）人脉

5）天赋

知识：能引发你改变的信息。比如C++多态和虚函数，是知识。

技能：《辞海》将技能定义为“运用知识和经验执行一定活动的能力”。运用C++中的多态和虚函数设计一个抽象工厂，解决对象创造问题，是技能。

关于知识和技能，可以用武功来类比，剑谱是知识，剑术是技能。你知道很多剑谱，但用不起来，这些剑谱对你就没什么价值。比如王语嫣通晓各种武术知识，但不能打，无法与人对决；再比如人们常说的书呆子，往往也是死读了很多书、储备了很多知识而不能灵活运用来解决问题。所以我们在盘点知识时，要多想想，这个知识是否可以运用起来创造价值。

盘点知识和技能时，可以从以下3个方面考虑：

• 技术

• 管理（领导力、组织、协调、制定计划、委派任务、辅导、目标整合、规划）

• 通用（演讲、写作、授课、沟通、共情、反馈、时间管理、目标管理、激励）

你可以使用思维导图把上述3类知识、技能梳理出来，保留导图原稿，每学到一种新的知识或技能，都可以把它们添加到思维导图上（我使用XMind社区版绘制）。

如图6-4所示是我的开发知识图谱。

从技术、管理、通用等3个类别中，挑出你喜欢的3个知识领域。我的如下：

• C++

• Qt

• 职业规划

• 授课

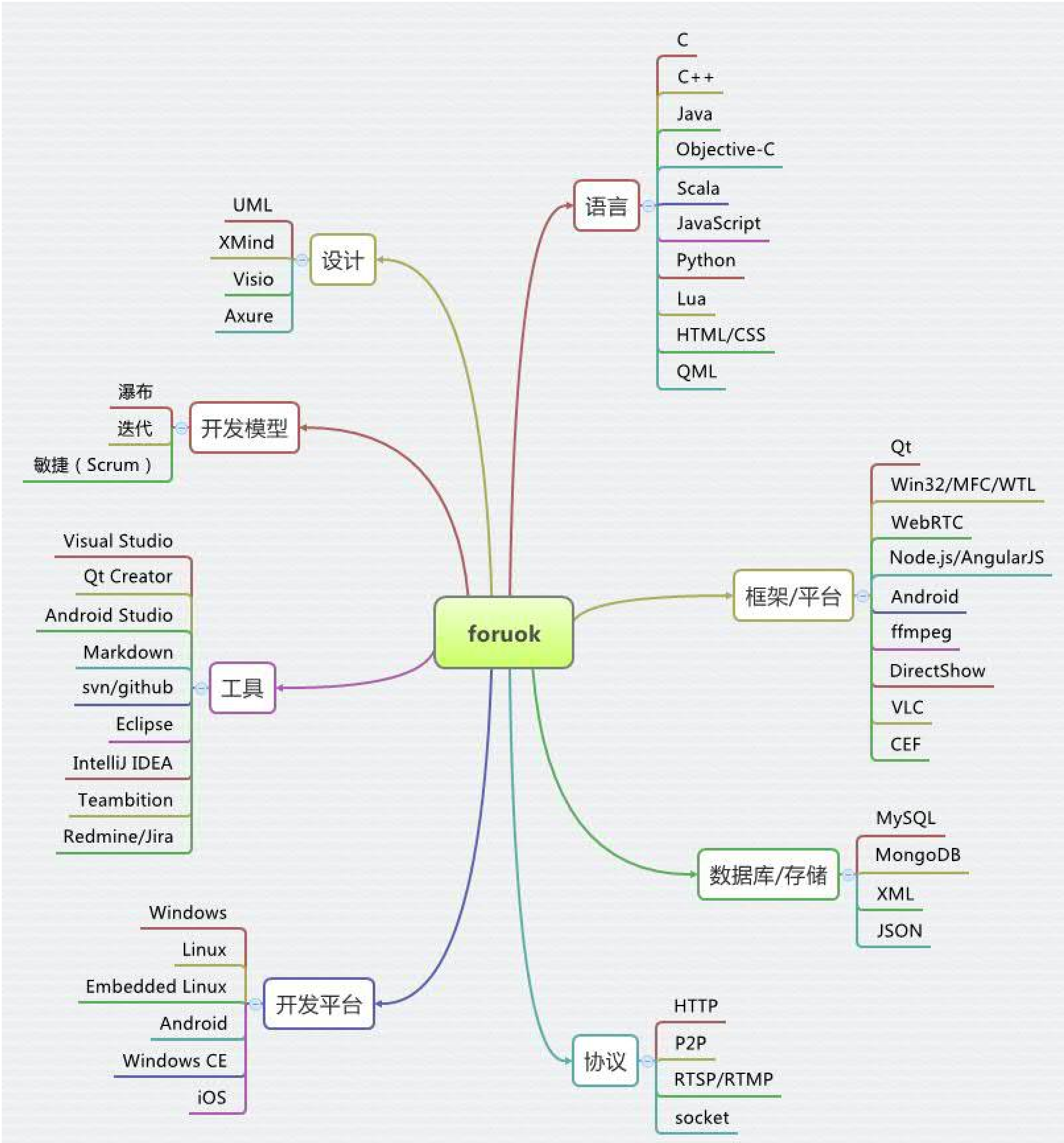


图6-4

挑出你想发展的3种技能。我的如下：

• 使用C++开发软件

• 授课

- 写作

现在，你应该有了一个知识技能清单，上面列出了我的3个知识领域和3种技能。接下来可以继续分析你的其他价值。

经历可以从以下方面挖掘：

- 学习
- 工作
- 生活

人脉可以根据关系来源进行盘点：

- 亲戚
- 同学
- 同事
- 技术好友
- 社群
- 同好

天赋有物理属性明显的，如：

- 身高
- 长相
- 声音

- 柔韧性
- 爆发力

也有软性的，如：

- 耐力
- 性格特征：细致、勤勉、敏感、专注、自律……

对开发者来讲，在简历中，通常会重点体现知识、技能、经历这3个方面。

在投递简历时，可以优先考虑人脉推荐的方式，它的成功率是海投简历的好几倍。

进入面试阶段后，软性天赋，比如专注、细致等，往往会大大影响对方对你的判断。你是不是他们想要的那个人，就在于如何展出你所具备的他们要求的各种软性特征。

需要注意的是，我们在这里盘点自己的商业价值时，是从个人视角出发的，没有考虑用人单位的视角。后面我们在撰写简历时，要转换视角，从用人单位的角度来看待你的商业价值。

6.4 寻找机会

有了工作特征清单和知识技能清单，就可以开始寻找机会了。这个过程又分为四步：

1）分析可能的职位

2）确认哪些公司提供这些职位

3）了解公司和职位信息

4）确定自己感兴趣的公司和职位

1.可能的职位

在6.3节“盘点你的价值和亮点”中，我们分析出了自己喜欢的3个知识领域和3种技能，形成了知识技能清单，从这个清单可以分析出可能适合我们的职位。

有些知识领域和技能可以直接关联到职位，比如我拥有“用C++开发软件”这种技能，就可以对应到“C++开发工程师”这个职位上。

有些知识、技能组合在一起可以对应到某个职位。比如我有讲课的技能，又熟悉Qt，那么就可以对应到“Qt培训讲师”这个职位上；再比如讲课技能和C++知识组合起来，对应“C++讲师”这个职位。

如果你熟悉软件开发相关的职位，就可以直接从知识、技能清单推演出职位。如果你不熟悉，还有一个更为直接的方法：在招聘网站（如拉勾、猎聘、智联等）上以你喜欢的知识或技能为关键词进行检索。

这么分析下来，与我喜欢的3个知识领域和3种技能相关的职位有这些：

• C++讲师

• Qt培训讲师

• C++开发工程师

• 职业规划师

• 职业规划培训师

• 职场作家

• 技术图书作者

2.哪些公司提供这些职位

得到职位清单后，就可以搜集你所在的地区有哪些公司提供这些职位了。

可以用下列方式：

• 常规招聘渠道（比如拉勾、猎聘、智联、领英等网站，BOSS直聘、脉脉等APP）

• 社群招聘渠道（比如论坛、QQ群、微信群等）

• 猎头（在你的联系人中，应当有几位做猎头的朋友；如果没有，那下一次有猎头和你打电话时，请记下他的联系方式，或者主动找猎头公司提供你的简历）

- 人脉（同事、同学、亲戚、朋友、工作中的客户、技术好友等）
- 你心仪或感兴趣的公司的官网

下列公司提供我感兴趣的职位：

- 北大青鸟
- 达内
- 华清远见
- Digia（迪智，我一直感兴趣的公司，Qt的拥有者）
- 西安诺瓦电子科技有限公司（一位做猎头的朋友提供的信息）
- 上海有智（USMART）网络科技有限公司（我与这家公司的创始人相熟）
- 广联达（猎头）
- 全时云商务（我最近服务过的公司）
- 东方网力
- 和利时
- 美亚柏科
- 北信源
- 绿盟科技

- 交大捷普（我爱人的朋友在这家公司做研发经理）
- 向阳生涯（我在这家学习的职业规划）
- 新精英
- 北森

在上面的清单中，我未做标注的公司，都是通过招聘网站检索得到的。

相信你也可以列出类似的清单。

3.了解公司和职位信息

你掌握的信息越全面、越多，你就越能做出有价值的选择。所以，在你列出公司清单后，就要下足功夫来了解公司和职位信息。

关于公司，了解下列信息：

- 公司产品所在行业
- 公司在行业内的地位
- 公司的产品（是什么、解决什么问题、谁在用、怎么用）
- 公司的薪酬水平
- 公司文化
- 公司位置
- 公司组织架构
- 开发部门在公司内的地位

关于职位，了解下列信息：

• 职位属于哪个部门

• 职位隶属于哪条产品线

• 产品或服务在公司的重要性（是拳头产品、新研产品还是内部支持性产品）

• 要求的知识、技能

• 工作职责

这里搜集的公司和职位信息是后面筛选公司的基础，现在只要尽可能广泛地搜集，先不要做评判。

搜集这些信息的途径有：

• 现在（或曾经）在目标公司工作的人（朋友、前同事、同学等直接人脉，或者二度、N度人脉）

• 官网

• 电话沟通

• 职场社交平台（脉脉、LinkedIn）

• 招聘网站（主要看招聘信息中的描述）

• 爆料网站（看准网http://www.kanzhun.com/、职友集http://www.jobui.com等）

• 薪酬报告

• 搜索

有一些小公司可能网上资料不多，也没有官网，只能从招聘信息中来揣测部分信息（比如公司的行业属性、产品等），这些信息也可能没有别的渠道佐证，难辨真假，此时你可以打电话过去问你想了解的问题。

稍具规模的比较正规的公司，都会有官网，可以从中了解到很多信息。

我们以广联达为例来说明一下。

广联达的官网地址是：http://www.glodon.com/。打开它的网站，鼠标悬停在浏览器标签页上，网站的标题会浮出来，其中有一个Slogon——数字建筑产业平台服务商。由此我们可以知道，这家公司所处的行业是建筑行业，细分方向是建筑行业中的数字建筑。

这里我们要先解释一下软件公司的行业属性，很多朋友容易搞混，以为软件公司的行业就是计算机软件。其实软件公司有两种行业属性，一种是计算机软件，另一种（也是更重要的）是这家公司软件产品所属的行业，也就是说，这家公司的软件产品解决什么领域的问题，它就具备这个领域所属的行业属性。有时一家公司可能具备多个领域的软件产品，那么它可能就有多个行业属性。比如广联达，它的主要产品是建筑类软件，那么它的一个行业属性就是建筑行业，它还有互联网金融产品，那么它就还有金融行业的属性。我们在看时，要看它的主要产品所属行业。

广联达在数字建筑领域是老大。

公司的产品，可以通过网站导航栏的“产品&方案”菜单来一一了解，如图6-5所示。



图6-5

公司文化、位置、组织结构等信息，都可以从公司的网站上了解到。

公司的薪酬水平，可以到职友集（http://www.jobui.com）、看准网（http://www. kanzhun.com/）等上面去查（有可能查不到），也可以参考行业薪酬报告，智联、猎聘、100offer、拉勾等，都会不定期发布薪资调研报告。

了解开发部门在公司内的地位非常必要，在核心价值链上的部门和职位，才会有比较好的发展前景。如果你到一家房地产公司做网站

开发，铁定是要被边缘化的，因为它是市场驱动型的公司，开发者在里面受重视的程度不如售楼人员。

职位的工作职责，所要求的知识、技能，一般都会罗列在招聘信息中。

你可以用Excel表格来保存你搜集到的信息，类似图6-6所示的这个表格。

每个Sheet代表一个公司，Sheet内最上方是公司信息，下部是职位信息，如果有多个职位，就继续往下延伸。

这个表格的名字叫“职业机会评估表”。

搜集公司和职位信息，整理出职业机会评估表，是简历优化过程中非常重要的一步，它的结果，一方面可以用于后面的筛选过程，另一方面，对于我们的面试、薪水谈判等过程也有很强的参考价值。

	公司信息	
1	**公司信息**	
2	行业属性	软件/建筑/建筑信息化
3	行业内地位	No.1
4	公司产品	……
5		……
6		……
7	薪酬水平	……
8	公司文化	……
9	公司位置	凤城十二路凯瑞大厦D座4层
10	公司组织架构	……
11	研发团队地位	重要
12	**C++开发工程师**	
13	部门	造价BG
14	产品线	广联达建设施工造价管理整体解决方案
15	产品重要性	重要
16	要求的知识、技能	C/C++ GUI开发，Qt，Qt Creator，Visual Studio MySQL、SQLite STL Socket、TCP、UDP、HTTP等网络编程
17	工作职责	1. 参与系统分析和设计，完成代码编写和测试。 2. 根据产品定义，完成各类开发文档的整理和编写。 3. 承接现有模块及产品, 新增、维护和优化现有功能及产品架构。

广联达 | 诺瓦 | 全时 | 和利时 | 东方网力 | 交大捷普

图6-6

由于这个过程耗时耗力，很多朋友懒得认真做，往往是看着职位名字和要求差不多，就投递了简历，结果要么通不过，要么面试时发现这根本不是自己想要的，甚至有的朋友在进了公司之后才发现双方不合适，最终给自己和公司造成大量的时间、精力浪费。

找工作是一个系统化的工程，只有做扎实、充分的准备，才可能用最低的成本找到最适合自己的机会。

4.确定自己感兴趣的公司和职位

有了职业机会评估表，接下来就是从中选出自己感兴趣的公司和职位。

我个人推荐遵循下面的顺序来筛选工作机会：

1）行业和产品

2）理想工作特征清单

选择处在上升期的行业，是保障你职业发展的基础。你进入到一个夕阳产业里面去，很难说会有好的发展。

对于开发者来讲，选择行业，其实就是选择公司开发的产品对应的行业属性，而非计算机软件或者互联网这层软件公司与生俱来的行业属性（前面介绍过软件公司行业属性的特别之处）。

举个例子，给煤炭、电力、钢铁、石油等产业做软件的公司，都会被我排除在外，因为它们服务的产业，要么产能过剩，要么在走下坡路，短期内很难看到希望，那么为其服务的软件公司，自然也会受到非常大的影响，难有好的发展，如果你在这样的软件公司中，很可能就没有特别好的前景。

我看好的方向也有很多，像教育、医疗、保健、安防、文化、娱乐、体育、电商、旅游、母婴、企业服务、游戏、自动化、智能制造、智慧交通等。

用你自己偏好的行业和产品筛过一轮之后，就可以使用第6.2节中整理出来的“理想工作的特征清单”来进行第二轮筛选：首先滤掉那些不满足你最想要特质的工作机会，然后筛掉不满足你次想要特质的工作机会……

这样一路筛下来，你的职业机会评估表就会变得非常简单，最后可能只剩三五个机会。

在我的理想工作特征清单上，第1位是通勤时间短，第2位是发展专长（包括C++和Qt开发、授课和写作），第3位是薪水。

如果我个人在西安找工作的话，首先就会把广联达筛掉，因为我住在高新区，它在城北凤城十二路，离我太远，开车要1个多小时，如果遇上堵车，可能得两个多小时，这是我不能忍受的。

这轮筛下来，我感兴趣的工作机会，就只剩下列几个：

- 有智
- 和利时
- 诺瓦
- 东方网力
- 全时云商务

这是非常短的清单！

6.5 招聘信息分析

要想让你的简历在3~5秒的时间内抓住目标公司简历筛选人员的眼睛，就必须要细致入微地研究招聘信息。

一份好的招聘信息应该包括：**工作内容、工作收获、工作成就、工作伙伴、工作认可与奖励、任职要求。**至少要包括工作内容和任职要求两项。

招聘信息一般由用人部门拟定，经由人力资源专员（HR）发布。用人部门的负责人在确认招聘需求时，一般是先看到要做什么项目（产品），然后根据这个项目（产品）的需要推导出要招募的人应该具备什么技能、经历。当他拟定招聘信息时，就会把脑海里这些技能、经历作为重点交代给HR，HR筛选简历时就按用人部门划的重点来做简单匹配。

所以我们在撰写简历之前，就要努力解码招聘信息，还原出用人部门的关键要求。

一般来讲，你可以从以下6方面提取招聘信息中的关键词：

1）学历

2）工作年限

3）知识

4）技能

5）项目经历

6）软能力

分析目标职位招聘信息的过程，逻辑上分为两步：

1）拆解关键词

2）对照自己的价值筛选匹配点

建议在做招聘信息分析时，先不管自己的情况，单练如何拆解关键词这一步。因为如果你在分解招聘信息时脑子里始终在观照自身，比较容易受否定信息的干扰，觉得这个自己没有，那个自己没有，几个没有之后，可能就会觉得自己不适合这个岗位，就放弃了。

当你能够客观、熟练地分析招聘信息后，再拿前面盘点出的自我价值和拆解出的关键词作比对，找到你和目标职位匹配的部分。

分步骤练习的次数多了，就熟练了，拆解和筛选两个步骤往往就会自然合一。

我分析过一个职位，如图6-7所示。

职位描述：

C++高级开发工程师

需求部门：开发部-西安研发中心

工作方向：产品开发

职位描述：

基于Qt框架进行相关桌面应用软件的开发。

对软件设计能力，编码能力，沟通能力，团队协作有较高要求。

1、全面理解业务需求，并负责根据需求，完成软件的原型、概要和详细设计

2、负责参与该系列软件的全过程开发及自测，保证代码内部的高质量。

3、负责该系列软件与其它关联软件的接口设计和维护。

4、在团队内开展技术传播，帮助其它团队成员完成C++技术的转型和学习。

任职要求：

1、4年以上 C++和windows桌面软件开发经验.

2、基本知识技能.

3、精通C++语言,熟练使用C++标准库.

4、MFC，QT框架至少熟悉一种.

5、熟悉图形界面的设计,熟悉常用设计模式,及算法.

6、有建筑类软件，AutoCad，3维图形相关软件编程经验者优先，有QT开发经验者优先.

7、掌握常用的OO设计原则和设计模式，并深入了解OO编程思想.

8、具备良好的编码习惯和良好的团队合作精神.

9、具备良好的沟通能力，做事踏实.

10、有完整的软件架构，设计经验者优先.

11、对新事物，新技术的洞察力强，敢于挑战高目标.

图6-7

我标注出来的，都是和我的个人情况匹配的关键点。

注意我们在分析招聘信息时，一定要关注“**优先**”条件。比如图6-7中波浪线画出的两个条件，都是优先条件，如果你具备，就会大大增加获得这个职位的概率。

如图6-8所示的招聘信息来自全时云商务西安分公司2017年的一个职位，大家可以用它来练练拆解关键词这个步骤。

>> 高级Java软件工程师

工作地点：西安

人数：2名

待遇范围：9-14k

岗位职责：

1. 负责BOSS系统框架的搭建及开发工作；
2. 负责BOSS产品的性能优化；
3. 能够独立负责某个子系统的开发

任职资格：

1. 计算机相关专业，5年以上JavaWeb开发经验；
2. 有扎实的Java编程功底，能熟练使用Spring、iBatis、hibernate等主流J2EE开发框架；
3. 了解SpringCloud、SpringBoot等微服务技术，有相关开发经验者优先；
4. 熟悉 BootStrap、jQuery、CSS、html5等前端技术及框架，参与过前端框架技术选型及搭建者优先；
5. 熟悉Nginx、Tomcat、Jboss等应用服务器，有相关服务器调优经验者优先；
6. 熟练使用MySQL数据库，对NoSQL存储技术有一定的了解，有数据库读写分离及优化经验者优先；
7. 熟悉应用程序调优、JVM调优，有高并发下系统调优经验者优先；
8. 良好的沟通交流能力，有较强的独立解决问题、自主学习能力，能快速熟悉并掌握新技术；
9. 有团队合作精神、能吃苦耐劳、有积极主动学习和探索精神。

图6-8

6.6 简历优化

谨记，在招聘过程中，企业是买家，我们是卖家，自己是商品，简历是宣传文案。我们要了解买家需要什么，在宣传文案中突出他想要的东西，这样才能打动他，让他感叹：喔喔，这就是我想要的！

当你筛选出招聘信息中和你匹配的关键词后，就可以有针对性地优化你的简历了。

注意，简历优化不是造假，而是忠实于你的个人情况，有选择地呈现对方想看的那部分。比如你精通Java、C++、Go三种语言，对方的职位要求Java，那么你突出Java，略去C++和Go，就是一种优化，因为这个过程去除了无用信息对简历挑选人员的干扰。

有一些朋友会捏造技能、工作经验和项目经历，比如只听说过Vue.js，就在简历上写“精通”，没做过项目管理，就在简历上说“有丰富的项目管理经验”。个人极其反对这种做法，因为这违反了“诚实守信”的基本原则。

简历优化主要分两部分：

1）技能评价栏

2）项目经历

1.技能评价栏的优化

先给大家看一个例子，如图6-9所示。

专业技能
精通 Java，能够熟练基于 SSH、SpringMVC+MyBatis 等框架的平台开发；
熟练的使用 C/C++语言进行服务器及客户端程序开发，熟练掌握 MFC、STL、ATL、多线程、GDI+和 API，熟悉 DirectShow、Socket 通讯、COM、ActiveX、DLL 等开发应用；
了解 Linux 编程环境及使用 GDB 进行代码调试；
熟练 ASP.NET、Winfrom、ASP 程序设计,熟悉 Web Service；
熟练使用 HTML、JavaScript、 DIV+CSS、AJAX、JSON、XML、JQuery 等技术；
熟练使用 SQL Server、MySQL 数据库，熟悉 Oracle 数据库；
熟练使用 VC6.0、VS2010、MyEclipse、source insight 等 IDE；
习惯使用 VSS、SVN、Git 进行团队协作开发；
熟练掌握 IIS，熟悉 Nginx 服务器的安装、配置应用；
熟练掌握 Linux 下的 Web 运行环境搭建；
熟悉相关技术文档的编写；熟悉产品原型工具 Axure RP 的使用；
熟悉常用 UML 建模工具 Visio、PowerDesigner、Rational Rose；
熟悉 TCP/IP、UDP、HTTP 协议；

图6-9

你看到这样的技能介绍内心有什么感觉？

多、杂、乱、晕，这是我的直观感受。然后我就会有一个评价：没有重点，缺乏针对性。最后我会推测这份简历的主人：没有用心，一份简历四处投递。

在我大学刚毕业及在职场的前几年，找工作时用的都是这种自我评价：罗列一大堆技能，这个精通、那个熟悉，好像自己上天入地无所不能。我的潜台词是：我很好，快收了我吧，你肯定不后悔。但是我忽略了“适合性”这一点，适合的，才是最好的。

所以，我们在优化技能评价栏时，要瞄准招聘信息中的关键词，列出匹配的三五点足矣。

2015年我写过一份通用简历，其中技能评价部分如下：

技能与评价

• 有7年部门管理经验，丰富的项目、团队管理经验

• 有7年嵌入式开发经验，在互联网电视机顶盒、车载娱乐系统、手持娱乐设备（MP3/MP4）等领域有成功经验

• 有丰富的软件系统架构设计经验

• 熟悉常见的设计模式，有丰富的面向对象设计经验

• 精通C/C++，熟悉Java、Shell，了解Python、Lua、JavaScript等

• 可熟练在Android、嵌入式Linux、Windows CE、Windows、Linux等平台下进行开发

• 熟悉Android/Qt(E)/MFC/WTL等GUI框架

• 熟悉DirectShow/GStreamer/MPlayer/FFMPEG/VLC/Vitamio等多媒体框架

• 熟悉网络编程和各种流媒体协议（HT・TP/HLS/RTMP/P2P/RTSP等）

• 博客：http://blog.csdn.net/foruok

• github：https://github.com/foruok

• 著有《Qt on Android核心编程》和《Qt Quick核心编程》

现在我们针对上一节给出的招聘信息来优化我的技能评价。

因为目标职位是高级软件开发工程师，招聘信息也没有特别体现对管理能力和经验的要求，所以我把管理经验拿掉，突出C++、Qt、设计模式、架构、技术传播等对方要求的要点。新的版本如下：

技能与评价

·精通C++，有9年开发经验，开发过C++11在线视频课程

·精通Qt，著有《Qt on Android核心编程》、《Qt Quick核心编程》，基于Qt开发过8个商业项目，写代码超过30万行

·熟悉GoF设计模式，撰写过设计模式专栏

·负责过100万用户流媒体系统的架构设计

·热衷技术传播，推动了Qt在团队和公司内的普及

这样的评价简洁、适合、有力。

2.项目经历的优化

在简历中梳理呈现你的项目经历时，请遵循下面4点：

·略去无关项目经历

·突出项目过程中用到的与目标职位匹配的技能

·描述你做的事情

·描述取得的成绩（显化、量化）

具体到撰写某个项目经历时，可以参考STAR原则：

·情境（Situation），对应到项目描述或项目背景

• 任务（Task），对应到项目目标或者你职责内的目标

• 行动（Action），对应到你做的事情

• 结果如何（Result），对应到项目的结果，尤其是你做的事情所产生的结果

下面是我个人的通用简历中的一个项目经历。

2009.05—2011.09，第一代互联网电视机顶盒。

项目描述：为解决用户在电视前看互联网视频而设计的机顶盒产品，聚合式视频导航，基于遥控器的首字母搜索，720p高清在线视频。

环境：Embedded Linux，Qt Embedded 4.5.1，恩智浦STB225方案，C/C++。

担任项目负责人兼项目经理，责任：

• 开发团队组建与管理

• 项目管理

• 硬件平台选型

• 软件系统结构设计

• 客户端业务流程代码实现

• 十字菜单EPG、影视墙实现

• 认证模块实现

• GStreamer input实现

下面是我针对“招聘信息分析”那节贴出的招聘信息运用STAR法则优化过的项目经历。

2009.05—2011.09，跨平台视频点播系统。

项目描述（Situation & Task）：面向电信运营商和零售市场的综合性视频服务产品，具有视频导航、搜索、点播、直播、天气、资讯、股票、教育等功能。产品形态有机顶盒、Windows客户端、Linux客户端等。

软件环境：Embedded Linux，Qt Embedded 4.5.1，Qt 4.5.1，C/C++。

硬件环境：全志、F20芯片方案、PC。

角色：项目经理、核心程序员。

职责与工作内容（Action）：

1）软件系统结构设计

2）基于Qt GraphicsView框架设计实现十字菜单、影视信息墙

3）基于Qt的客户端业务流程代码实现

4）基于Qt实现认证模块

5）团队组建与项目管理

业绩（Result）：

1）开发了稳定、性价比高的互联网机顶盒产品，打开了电信市场，销量超过30万台。

2）在团队里推广Qt，形成了技术积累。

3）提供Windows及Linux客户端，为客服、运维、售后提供了强有力的支撑。

请假想你是用人部门负责筛选简历的开发者，结合前面的招聘需求，问问自己，你会喜欢哪个版本。

6.7 检验简历优化效果

要检验你简历优化的效果，有如下3种方法：

1）设想自己是HR或研发部门的程序员，根据招聘需求来评估简历

2）请他人担任简历筛选人员，让他根据招聘信息来评估你的简历

3）选取一些公司的职位，优化简历，投递实验

前两种方法比较容易理解，也容易践行。第3种方法，实际上是把检验效果这个动作融入到了求职过程中。为避免你不合格的简历被心仪的公司刷掉，失去再次投递的机会，建议将你感兴趣的公司分成3类：

1）A类，最想去，这类公司就是自己的理想目标

2）B类，想去，公司有几点吸引自己的地方

3）C类，一般想去，可以试试看

然后按照C、B、A的顺序投递简历。这样做的好处是：

1）检验简历优化水平

2）积累面试经验

3）了解行业、产品、薪酬等信息

4）可能发现意外的机会——也许你开始不很想去的公司，了解后特别想去

第3）种方法可能会额外增加有些公司的招聘成本——因为你即便拿到了Offer也不会去。所以，如何做，要不要这么做，遵循你的内心选择。

6.8 如何提高简历投递成功率

根据前面的分析，要想提高简历投递的成功概率，遵循下面的流程将非常有帮助：

1）确立求职目标

2）梳理知识、技能、经历并记录在案，形成基础简历

3）筛选招聘信息，选择匹配自己目标的公司和职位

4）针对每个招聘信息进行分析，提取关键词

5）根据关键词，结合基础简历，优化技能描述和项目经验，生成一份有针对性的简历

根据我自己的经验，一天可能只能完成3份左右的简历投递。因为针对每一个职位生成一份有针对性的简历，可能会花费1到2个小时时间，有时甚至更长。不过，花再多的时间都是值得的，因为这种优化将大大提高简历通过的概率。

第7章 如何在跳槽时获得想要的薪水

选择工作有两种因素：保障因素和激励因素。

保障因素包括地位、薪水、安全保障、工作条件等。只有在保障因素到位的情况下，人才能正常地、踏踏实实地工作。

薪水是最重要的一个保障因素，所以每个人在找工作时，都想要拿到理想的薪水——哪怕你为了个人未来职业发展选择降薪求职，还是会有一个心理预期。

那么，怎样才能在跳槽时获得自己想要的薪水呢？

我从薪水谋划的角度总结了跳槽的流程，如图7-1所示。

这个流程呈现了非常关键的一点：**薪水不是谈判出来的，而是取决于双方的匹配度。**匹配度在薪水谈判开始之前就决定了，在面试时我们能做的是想办法呈现自己和职位的匹配度，具体谈薪水在整个环节中反倒占不了多大的比重。

所以在本章接下来的篇幅中，我们会以这个结论为出发点，了解薪水是由哪些因素决定的，看看我们能控制（影响）哪些因素，学习怎样为薪水谈判做准备，介绍薪资谈判的6个秘密，让每个人都能从整体的角度考虑如何获得想要的薪水。



图7-1

7.1 决定薪水的7大因素

薪水由下列因素决定，其简单描述如图7-2所示。

- 市场行情
- 个人基于自我价值和市场行情的自我评估
- 公司的薪酬结构
- 公司基于需求和行情的评估（产生公司的预期薪水范围）
- 个人价值与公司需求的匹配度
- 公司需求的紧迫性
- 公司需求的重要性

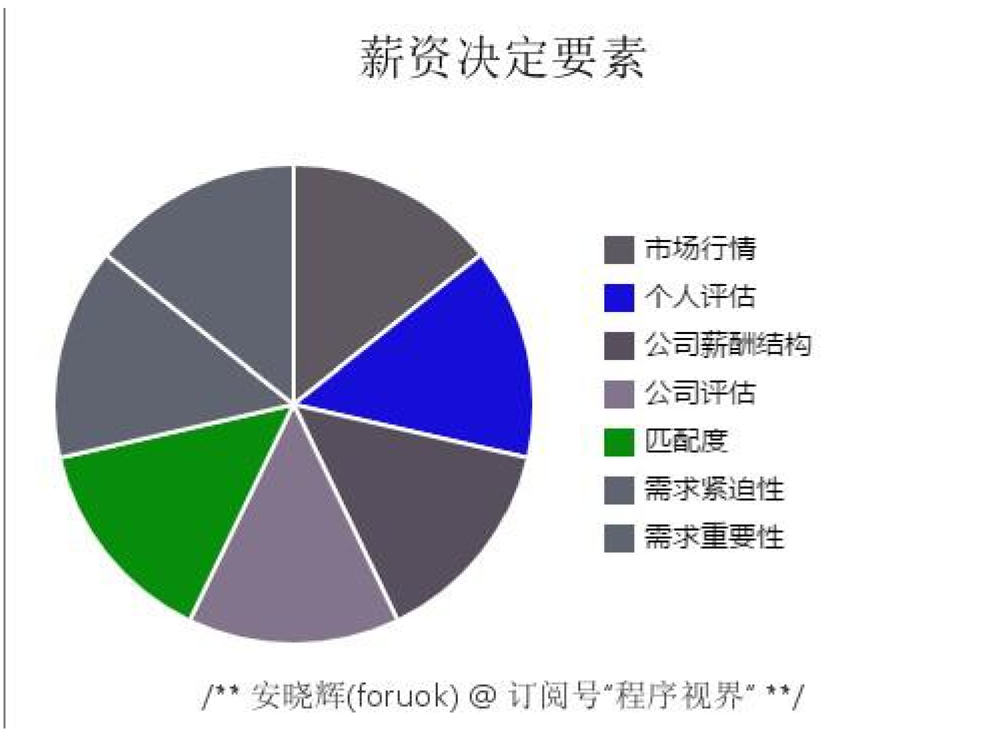


图7-2

1.市场行情

从一定角度看，开发者是卖家，企业是买家，开发者的知识、技能、时间等组合在一起形成各式各样的虚拟商品，开发者和企业在市场上自由交易，商品的价格遵循市场行情。

市场会有波动，可能某段时间某一类商品行情特别好，比如2014年、2015年的安卓开发，也可能某一段时间该类商品的行情又变得特别差，比如2016年、2017年的安卓开发。从整体上看，作为卖家的开发者，必然受市场行情左右，个人很难超越行情——除非你拥有非常稀缺的价值，获得了定价权。

2.个人自我评估

市场行情是平均水准，自我评估时需要判断你自己的位置，是高于还是低于平均水准，相应地，把自己的价钱上浮或者下浮一个比例，比如30%。这样，你的卖价就比较合理，既不会因为自我认知偏低而遇到刻意压价的企业而贱卖，也不会因为开价太离谱而错过本该属于你的机会。

有一个很有意思的现象，每个人都应该知道：为了吸引到想要的高素质员工，或为了吸引到足够数量的员工，很多雇主会提供高出市场行情的工资。这一般是因为企业认为员工获得了高于市场行情的薪水，会有比较强烈的经济动机来积极工作。从这一点来看，你在自我评估时，即便自己的能力是平均水平，也可以给自己适当加价。

3.公司薪酬结构

很多公司在设计薪酬结构时，采用宽带机制，为每一级别的工程师，设定一个薪水范围。当面试官认定你的能力和某个职级匹配时，你的薪水就很难超过这个级别的上限。HR往往会从某个级别的下限薪

水和你谈，如果你个人没有明确的自我评估，就会拿到一个比较低的薪水。

我爱人2006年回西安时，就遇到了这样的问题，被定了她所在职级的较低薪水，为此苦恼了四五年，后来因为领导觉得她工作出色，晋级时给了一个非常大的薪水涨幅，才拉平了和其他同级别同事的薪水差距。

4.公司评估

市场行情会对公司的薪酬宽带起到修正作用，导致公司在发布某个岗位时，会随行就市重新评估，最终给出的薪水范围，可能和公司内部对这个职位的定价不太一致。

比如某公司内部对有3年Android开发经验的程序员的薪资宽带是15000~22000元，而2016年、2017年Android开发行情很差，它给出的薪水范围可能就是15000~18000元。再比如2017年人工智能很火，很多创业公司就会开出高薪来吸引人才。

5.匹配度

从买方的角度看，你越匹配他们的需求，他们越愿意给你提供更高的、接近薪酬宽带上限的薪水，假如他们认定你就是那个人（You are the one!），薪水谈判时很可能会再给你额外的激励。而如果你凑合可以用，那么他们就会给你同级别的最低薪水，并且很难在薪水谈判时再做让步。

所以，你是否和目标职位的需求匹配是非常关键的，这决定了你能拿到多少薪水，也决定了你是否有议价权。

6.需求紧迫性

公司越急着招你进去顶位赶项目，越可能给你高出市场行情的薪水。假如公司发布的某个职位，仅仅是为将来的人才储备做预算，那么它往往不着急，会慢慢挑，直到挑到超出预期的人，才会给Offer。

7.重要性

岗位重要，薪水就高；反之，薪水就低。所以开发者在找工作时要了解这个职位做什么产品，在公司内的地位如何。一般来讲，与公司当前牛产品相关或者与公司未来战略方向相关的岗位，薪水偏高。

7.2 我们能直接控制哪些因素

仔细研究决定薪水的7大因素，你就会发现，你能直接控制的实在有限：只有**个人自我评估**和**匹配度**。

所以，我们在跳槽时，就应该把精力花在这两方面，也只有花在这两方面的时间和精力，才会有真正的回报。同时，围绕着这两点所做的工作，影响力还可能扩散，带动其他因素产生积极的变化。

《高效能人士的七个习惯》一书中提到了影响圈和关注圈理论，我把决定薪水的7大因素标作了归类，绘制了图7-3，可供参考。

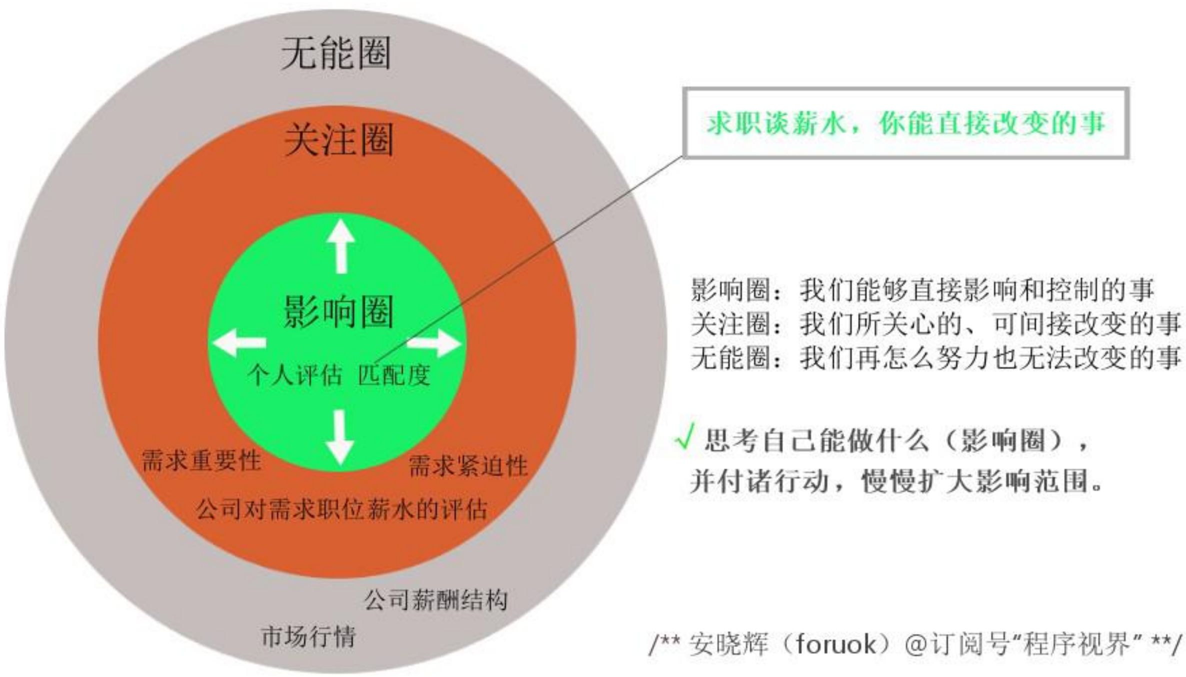


图7-3

个人评估基于你对自己价值和亮点的挖掘，你能在自己身上找到3项可以清晰描述的技能（知识）并能用实际的项目经历为其背书，就能打动很多公司。类似H.265或者VP9编码、推荐算法、OpenGL、Unity

3D、图像处理、流媒体协议、VoIP等这些，都能为你带来不错的竞争力。

当你明确了自己的价值点之后，就可以给自己定价了。比如你觉得自己对图像处理算法的掌握超越一般的工程师，那么就可以到拉勾、猎聘等网站上搜索相关职位，统计薪水范围，计算平均值，了解行情，然后做出自己的薪水预期。

匹配度这个因素的核心就是“知己知彼”。

“知己”这部分，在个人评估的过程中完成。

“知彼”这部分，主要是了解公司、产品、职位三方面的信息，我们在第6章“简历优化指南”的6.4节“寻找机会”中有详细的介绍，强烈建议翻回去看看。

你对目标公司、产品、职位了解得越多，在面试过程中就越能展现出你的态度和专业性，此处花一天、两天、一周甚至更久的时间都是值得的。

当你了解了公司所在行业、公司的产品、职位的要求后，你就能有的放矢，选择和你匹配的机会，选择你和这个机会匹配的点来优化你的展示（简历、面试等），这样双方的匹配度就会提升。

像下面这些面试中经常被问到的问题，都是匹配度相关的，你有了自我定位，了解了公司、产品、职位后，都可以回答：

- 你了解我们公司吗？
- 你为什么选择（应聘）我们公司？
- 你认为我们为什么要选择你？

7.3 如何为谈薪水做准备

薪水不是谈判出来的，而是取决于双方的匹配度。所以如果想要在跳槽时获得你想要的薪水，从一开始就要瞄准匹配度，为之做准备。

我总结了一个框架，分为8步，大家可以对照着它来为你的跳槽做准备。

1.明确你想要的工作特征。

开发者的求职方向，大多数还是落在软件开发相关的岗位上。如果跳出软件开发，就是转型了，我们会在下一章中介绍。

在这一步我们不考虑具体的软件开发职位，而是考虑一些软性的工作特征，比如有挑战性、受尊重、氛围好等，把它们找出来。具体参见第5章的5.2节“什么时候跳槽好”下面的需求供给分析法。

2.盘点你的价值点。

知识、技能、项目积累、行业积累、软能力……这些都可能成为你的价值点，请参考第6章的6.3节“盘点你的价值和亮点”，分析出你喜欢的3个知识领域和愿意发展的3种技能。

3.寻找与你价值点匹配的工作机会。

请参考第6章“简历优化指南”的6.4节“寻找机会”。

迄今为止的3点，是匹配度的基石，做好它们，你会在大方向上获得匹配度保障。

这一步尤其重要，你会了解到公司、产品、职位等信息，形成类似图7-4中所示的**职业机会评估表。**

这张表里的信息，在你优化简历、准备面试时，会很有帮助。

4.了解市场行情。

有两种途径可以了解市场行情：招聘网站职位信息分析和薪酬调研报告。

现在很多公司在招人时都会给出薪水范围，这可以提升双向选择效率。

基于这一点，我们可以在招聘网站按地区搜索特定职位，搜集薪水范围，进行统计分析，得出某个地区某个职位的行情。

1	**公司信息**	
2	行业属性	软件/建筑/建筑信息化
3	行业内地位	No.1
4	公司产品	……
5		……
6		……
7	薪酬水平	……
8	公司文化	……
9	公司位置	凤城十二路凯瑞大厦D座4层
10	公司组织架构	……
11	研发团队地位	重要
12	**C++开发工程师**	
13	部门	造价BG
14	产品线	广联达建设施工造价管理整体解决方案
15	产品重要性	重要
16	要求的知识、技能	C/C++ GUI开发，Qt，Qt Creator，Visual Studio MySQL、SQLite STL Socket、TCP、UDP、HTTP等网络编程
17	工作职责	1、 参与系统分析和设计，完成代码编写和测试。 2、 根据产品定义，完成各类开发文档的整理和编写。 3、 承接现有模块及产品, 新增、维护和优化现有功能及产品架构。

广联达 诺瓦 全时 和利时 东方网力 交大捷普

图7-4

如图7-5所示是在拉勾网（https://www.lagou.com/）上针对北京地区搜索“图像处理”关键词的结果。

如图7-6所示是在猎聘（https://www.liepin.com/）上检索的结果。


图像处理工程师 [海淀区] 1天前发布
25k-50k 经验3-5年 / 本科
视觉
算法
模式识别
图像处理开发工程师 [西北旺] 09:30发布
20k-30k 经验5-10年 / 本科
专家
资深
高级
算法
图形处理
图像处理工程师 [西三旗] 1天前发布
10k-20k 经验不限 / 硕士
算法开发
图像处理研发工程师 [上地] 1天前发布
15k-30k 经验1-3年 / 硕士
高级
中级
C/C++
C++
深度学习
模式识别

图7–5



图7-6

这种方法比较花费时间和精力，但只要你愿意严肃对待找工作这件事，就可以做到。

第二种方法是阅读薪酬调研报告，像 100offer、智联等都会发布薪酬报告或包含薪酬调研信息的行业分析报告，你也可以用搜索引

擎，以“互联网人才流动报告”“开发者薪酬报告”或者“互联网薪资调查”为关键词搜索。

这种方法不花什么力气，但拿到的往往是较大方向的行情，比如工作5年的Java工程师的年薪之类的。如图7-7所示是来自100offer 2016年的数据。

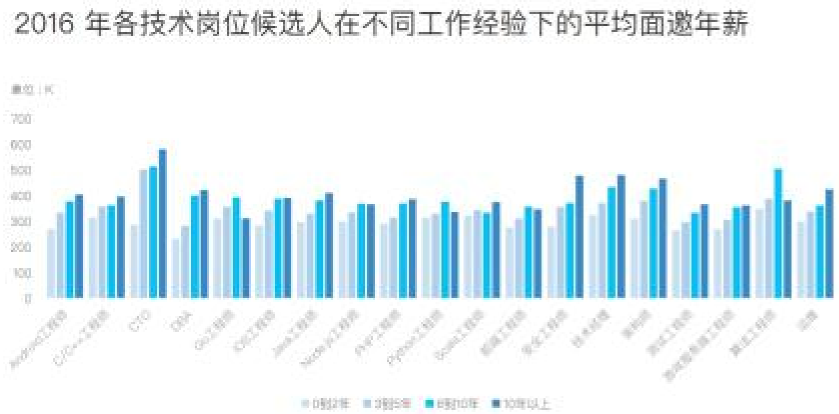


图7-7

5.自我评估

自我评估时，以市场行情为基准，上下浮动20%~30%即可。比如市场行情是10000元/月，那么你据此可以评估出最低薪资8000元，满意薪资13000元，那么你就有了8000元、10000元和13000元三个数据。

假如你认为自己能力远超平均水平，上浮50%，或者翻番也是可以的，但是怎么判断你的能力是高于还是低于平均水平呢？

有两种方法可以测量你的水平：

1）在你所服务的公司，你的技能水平是否在前20%？

2）在公开环境中去感受，比如在互联网上，与你所用技术相关的博客、文章、电子书所讨论的内容，你是否精熟或者觉得自己能轻松

搞定？再比如你所用技术相关的图书，其内容你是否觉得很简单？是否可以自己分享给别人并且让别人也明白？

如果两种方法的结果均为“是”，那么你的技术能力就极有可能远超平均水平。

自我评估的实质是管理自己的期望。当你明确了最低薪资、市场薪资和满意薪资之后，就可以根据实际的求职过程来决定是否接受一家公司给出的Offer。

比如你想往人工智能方向转型，但之前又没有任何经验，那么当你和公司在薪水上不一致时，为了未来的发展，只要Offer提供的薪资不低于你心中的最低薪资，就可以尝试。

再比如你提前做了评估，设定你的满意薪资是25000元，那么可能你遇到一个产品方向不错又愿意提供25000元薪水给你的公司，就会毫不犹豫地接受它的Offer。

6.搜集目标职位的可能薪水范围

第4点（了解市场行情），介绍了通过拉勾、猎聘等招聘渠道评估薪水的方法，实际上你在搜集某个公司的特定职位时，也可以用这种方法。因为很多公司为了提升效率，会把薪水范围写在招聘信息中。这是最直接的方法。

但招聘信息中的薪水范围往往很“宽”，比如一家公司招募图像处理算法工程师，标明年薪40万元~80万元，上限是下限的2倍，此时你可能就会有疑问：这个范围是不是专门写来吸引人的而实际上不可能给到？

我们不排除有些公司会这么做，但大部分公司，标注出来的范围上限都是可以给到的，甚至可能发给你薪水超过上限的Offer。

2015年年底创业失败后我决定重回开发岗位，当时有个猎头提供了一个来自广联达的机会（这也是我在前面会用广联达做示例的原因），并且告诉了我对方给出的月薪上限。我经历了三轮面试后，拿到了比猎头给出的月薪上限高4000元左右的Offer。

所以我觉得你要相信公司在招聘信息中给出的薪水范围，只要他们觉得你就是那个人（You are the one.），你就可能拿到接近上限的Offer。

除了招聘信息，还有如下渠道可以尝试：

1）向身边的在目标公司工作过的朋友、同事、同学等打听

2）在看准网、脉脉、职友集等平台搜索

3）在技术社群（QQ群、微信群等）中询问

找工作是非常重要、非常严肃的事情，可能影响你未来几年的幸福感，而薪水会严重影响你对一份工作的感受，所以在开始之前，多花些力气搜集薪水信息，非常必要，也非常值得。因此我们要打破“不好意思求人”“担心别人觉得自己太在意钱”这种思维定式，关系到我们将来发展和幸福的事情，为什么不提前问清楚？你要对自己负责哦！

7.有针对性地优化简历

第6章详细介绍了如何优化技能评价栏和项目经历。

8.搜集常见的面试问题，精心准备

可以从技术和非技术两方面来准备。

现在市面上有各种“程序员面试宝典”之类的图书和资料，网上搜搜，或者上京东买几本，结合你前面整理出来的职业机会评估表，找到相关的技术问题，多琢磨琢磨，基本上就可以应对宽泛的技术面试问题。

对于你做过的和目标职位相关的项目，多了解一下用到的技术，比如你用了NIO、Redis或者WebRTC，就搞明白你为什么选择它们，了解一下它们适用于什么场景、能解决什么问题、可以带来什么好处，再了解一下其原理，有可能的话，看看源码或者设计文档，基本上就可以应对针对项目经历衍生出的技术问题。

对于非技术类的问题，比如我们在前面提到的“你为什么选择（应聘）我们公司？”“你了解我们公司吗？”“你认为我们为什么要选择你”，再如“自我介绍”“你为什么离开上一家公司？”“你有什么问题要问的吗？”，都可以在了解目标公司、产品、职位后，提前模拟演练。

这类问题，直接通过百度或者知乎站内搜索，就能找到非常多的信息，可以结合自己的具体情况拟定答案，写下来，放到有道云笔记或者印象笔记中，经常看看。

7.4 薪资谈判的6个秘密

1.谁来拍板给你发多少薪水

不是HR。虽然HR会负责和你谈薪水，但决定你的薪水的不是他们。

能决定给你发多少薪水的，是招人的部门领导（研发经理、部门负责人）或者他的领导（VP、CTO、研发总监、技术副总等）。所以，你应该了解面试流程有几个，都是谁在面试你，要认真对待每一轮技术面试，这些面试的表现，将最终决定你能拿到什么样的薪水。如果技术老大觉得你很合适，当场就可能拍板要你，给你薪水宽带的上限，然后让HR和你谈Offer。

2.谈钱不伤感情，不谈才伤

有的朋友和我一样不太习惯谈薪水，觉得谈钱伤感情，影响后期成为同事之后的关系。但实际情况是，不谈薪水才容易伤感情。比如对方给了你一个较低的薪水，你进了公司就会发现身边同等资历的人薪水都比你高，你就会严重不平衡。

3.薪水是可以谈的

很多公司给每个职位都有一个预期的薪水范围（宽带），最低多少，最高多少，我们要谈的，就是这个范围。

在招聘时，公司希望用较少的钱招到合适的员工。所以很多人看到招聘信息中的薪水范围都会笑一笑：最低值才是真实的。但实际上，上限也是可能的，你要努力得到他们能提供的最高工资。

这其中的关键就是：瞄准你和目标职位的匹配度精心准备，让面试官觉得你就是他们想要的那个人。

4.绝不要主动谈论薪水，除非他们问你

如果对方觉得你就是他们要的那个人，他们就会和你谈薪水。可能在第一面，也可能在第三面。不管如何，在发Offer之前，他们一定会和你谈。

有的公司只想找能干活、要钱少的员工，一上来就问你期望的薪水，你可以回答：

“我想在你们决定录用我时讨论薪水会更合适。”

“我很愿意回答，但我想先了解一下，这个工作都需要做什么。”

“我期望的薪水是30000元~50000元/月”或者“我期望的薪水是30万元~50万元/年”，注意要给一个范围，不要给固定数字。

5.千万别先提数字

先提薪资数字的人，往往会输。

如果对方让你先提数字，你可以这样回答：“你们对这个职位的薪水一定是有所考虑的，我想先听听那个数字。”

如果对方执意让你说，就说一个范围（这时此前的调查工作就有用了），这个范围要夹住他们的薪水范围上限，类似于图7-8所示的这样。

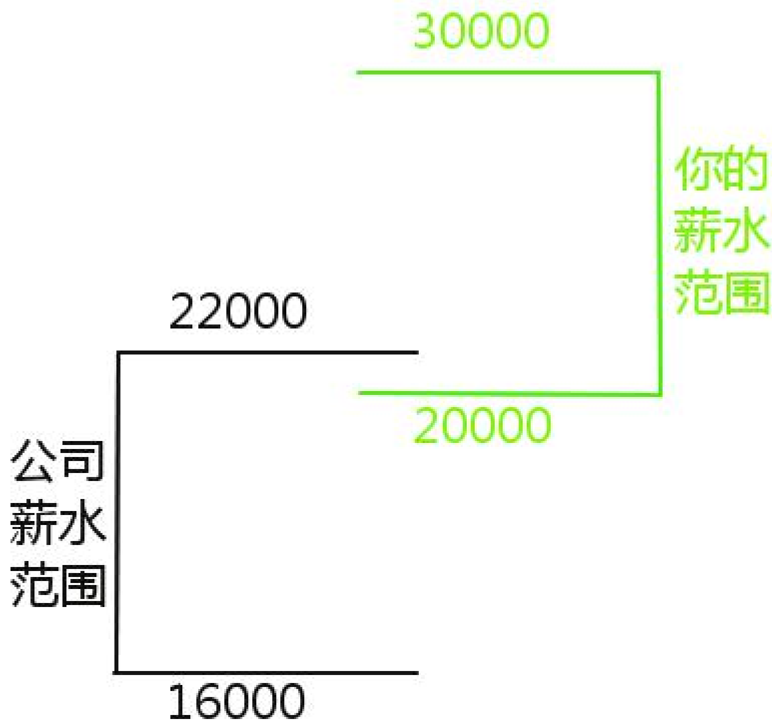


图7-8

6.福利

搞定基本工资后，就要问问他们有什么福利。

要清楚常见的五险一金，缴纳基数和比例。这会有很大差别。比如你谈的月薪20000元，公司会帮你缴纳12%的公积金，若基数是3000元，那么每月公司就帮你缴纳360元，如果公司给你的公积金缴纳基数是15000元，那么公司每月就帮你缴纳1800元，差别很大。

年假、婚假、陪产假、事假、病假等，都可以问。

绩效、奖金、期权、股票，也可以谈。

注意，要事先琢磨一下你最看重哪个福利，比如公积金、医疗保险和年假。如果你什么都一板一眼地去抠，有可能给对方留下负面印

象。

7.5 什么时候可以降薪求职

在第5章“跳槽8问”的5.2节中的“需求供给分析法”处，介绍了工作的43种特征。我们在找工作时，要先明确自己想要的工作特征，列出自己的理想工作的特征清单。比如我看重的就是：

1）通勤时间短

2）发展专长

3）薪水

4）创造性与自我表达

5）智性刺激

弄明白你想要的工作应该具备什么特征非常重要，它们是你选择一份工作的依据，能指导你理性地做出选择。

强烈建议你在跳槽之前，梳理出自己最看重的几个点，像上面那样排排序，优先级最高的放在第1位。

当薪水之外的某个因素被你排在了清单的第一位时，你就可能会选择降薪求职。

2013年时，一位来自华为手机部门的开发者应聘我负责的研发部门的职位，面谈时我问他“华为薪酬待遇在西安是最好的，你为什么要离开华为？”他告诉我说他在荣耀团队，每两个星期都要发版，每天晚上都要加班到11点左右，根本没时间照顾他怀孕的妻子。

那么这位朋友当时的理想工作特征清单可能是这样的：

1）轻松不加班

2）通勤时间短

它们指向了“照顾怀孕的妻子”这个更重要的需求，所以他选择了降薪求职。

2005年我在西安大唐电信做售后技术支持，月均收入6000多元，后来降薪转型做软件开发，月薪为2500元。当时我看中开发类工作的两点：

1）不出差

2）发展空间大

上面两个例子，对应了两类降薪求职的情况：

1）想要稳定、轻松的工作，可以照顾家里

2）想要完成转型，缺乏相关经验

7.6 薪水与幸福成正比吗

《小狗钱钱》一书中，陶穆老太太解释了金钱和幸福的关系：

要想过更幸福、更满意的生活，人就得改变自身。这和钱无关，金钱本身既不会使人幸福，也不会带来不幸。金钱是中性的，既不好，也不坏。只有当钱属于某一个人的时候，它才会对这个人产生好的影响或坏的影响。钱可以被用于好的用途，也可以被用于坏的用途。一个幸福的人有了钱会更幸福；而一个悲观忧虑的人，钱越多，烦恼就越多。

金钱会暴露（注意这个词，暴露）一个人的本性，金钱就像一个放大镜，它帮你更充分地展现出你本来的样子。好人可以用钱做很多好事。而如果你是盗贼，那你很可能会把钱挥霍在一些蠢事上。

我的一个基本的看法是：金钱在你贫穷（所谓贫穷，指的是你为了买一件生活必需品而必须舍弃另一件这种状态）时会带来比较直接的幸福感，而一旦你的生活离开了贫穷状态，金钱的增加可能会降低幸福感——因为你为获得更多的金钱而牺牲了更多东西，比如时间、兴趣爱好、陪伴家人和朋友。

所以薪水与幸福并不成正比，它们之间有一个临界点。这个临界点就是“获取更多的金钱会削弱你的幸福感”那个时候。

第8章 转型

有一天你发现了更喜欢的事情，画画、写作、演讲、旅游、教育、音乐、育儿、营销……恨不能肋生双翅立马飞去……

有一天你厌倦了“设计、编码、集成、解Bug、发布”这样的循环，再也无法多做一天，可是手拎键盘四顾茫然……

有一天父母或者妻儿对你说，别再做开发了，连家都顾不上……

有一天你体力不支，技术落后，难以胜任，公司觉得你的性价比不再合理，你不得不黯然离场……

以下是4种典型的转型：

1）为了明确喜欢的方向而告别开发岗位

2）厌倦了开发，但不知道离开可以做什么

3）被家人劝退

4）被公司辞退

如果你符合2）、3）、4）这三种情况之一，或者你只是做着做着感到迷惘了，失去了意义感，不知道该不该做下去，那么请留下来。

本章会带你重新审视一下自己是否还喜欢开发工作，看看程序员转型有何实际的困难、转型有哪些分类，再了解一下开发者常见的转型方向，还会提供一个“人事物模型”来辅助你寻找职业方向，最后还会提供一个转型实践框架，帮助你把转型落地。

8.1 你真的不再喜欢开发工作了吗

是否喜欢是主观的，是感情、情绪方面的，是一种感受，通过了解自己的感受来判断自己是不是不喜欢开发工作了，是比较自然的方式。

你可以从两个角度来体会：

1）工作时的感受

2）对时间的感觉

要了解自己对一件事情的感受，可以通过提问来探索自己做这件事时的感觉，然后分析一下就能知道是否喜欢。

1.工作时的感受

下面是我总结的和工作本身相关的几个问题，算是一个自我测试框架，通过问自己这些问题，你可以判断自己是否真的喜欢软件开发工作。

1）看到代码是否有“似曾相识燕归来”的温暖？

2）隔一段时间不写代码，是否会充满怀念，有想打开IDE写点什么的冲动？

3）是否经常有这样的时刻：看着自己的代码，有种“相看两不厌，唯有敬亭山”的喜悦？

4）有没有那么一些时候，你看着自己的代码，会不自觉地想：这里或那里改改是不是更好一些？

5）当你看到令人眼前一亮的APP或网站或其他软件时，会不会发出“要是我来做该怎么做”之类的问题？

6）你有没有想让别人阅读你的代码的冲动？

7）你有没有读别人代码的冲动（想看到更好的代码）？

8）别人指出Bug、错误或设计瑕疵，你会生气、拒绝还是接纳、感激？

9）修复一个Bug，你是为这个Bug被解决掉而高兴多一些还是为你的代码（软件）更完美而高兴多一些？

10）听到新语言、新框架、新系统、开发者大会等相关的消息，你是很想了解还是懒得搭理？

11）有技术大咖在你身边出现时，是想去结交还是懒得理他？

12）看见别人的烂代码时，你是吐槽其真烂然后绕过还是想怎么改好？

13）看见别人的优秀代码，会不会羡慕，会不会想“要是我也能写出这么漂亮的代码就好了”？

14）当你完成一个模块、功能、系统，解决一个问题时，是有“成就感”还是有“终于交差了”的感觉？

15）想到你开发的软件可以帮助别人解决问题带来好处你是否感到期待、兴奋？

16）你是否想建立属于自己的软件资源（比如工具、类库）？

17）你是不是像蜜蜂一样总是把看到的与软件相关的好东西收藏起来？

2.对时间的感觉

时间是一个棱镜，可以折射出你对某件事情的感觉。问问自己下面几个问题吧：

1）写代码让你觉得时光飞逝如箭还是一秒犹如一万年？

2）当你回顾一天、一周、一月的工作时，是否经常后悔，觉得自己应该在开发上投入更多的时间？

3）当你回顾一天、一周、一月的工作时，是否经常后悔自己在开发上花费了太多时间？

4）你觉得花费在软件开发上的时间值得吗？对你有什么意义？

8.2 程序员转型的难处

如果分析之后，发现你真的不喜欢开发了，那么可能转型就是必然的了。在转型之前，了解一下可能面临的难处，能让我们有更好的准备。

常见的难点有5个：

1）路径依赖导致的隐藏假设

2）薪水落差

3）缺乏技能

4）他人的期待

5）对不确定性的抗拒

1.路径依赖导致的隐藏假设

我每天开车到单位，都走“科技路→沣惠南路→科技六路→团结南路→科技七路→丈八东路→丈八西路→丈八二路→锦业一路”这样一条线，下班再反向回来。不管路上是否堵车，我都这么走。有时跑个高速后回西安，也会从高新出口下，然后再回到这条线上。

这就是我的路径依赖，当我走上这条路时，就觉得踏实、安全。

对于从事软件开发的我们来讲，也有这么个路径依赖：会习惯性地走在开发这条路上。哪怕正在慢慢丧失竞争力，哪怕公司摇摇欲坠朝不保夕，还是会这么走着。即便哪一天不得不重新找工作，还是会第一个想到：换家公司继续做开发。

这是人的天性：长时间做某件事情，就会对它产生依赖、认同。一旦我们习惯了某件事，就会被它植入相应的隐藏假设——你必须做这个，进而在我们必须做出选择时影响我们的选择。

以开发为例，很多人心中都有了这样的隐藏假设：如果你做了开发，就只能一直做开发。

正是因为这种隐藏假设遮蔽了我们的心眼，我们在转型时很少看见开发之外的可能性。

这种路径依赖和隐藏假设牢牢地束缚住我们，一方面是因为我们不愿意放弃经年累月辛苦积累起来的价值，另一方面是因为我们低估了自己的潜能，高估了在新领域取得成绩的难度。

关于在开发经历中积攒起来的价值，在转型时并不会归零！你在这条路上积累的经验、阅历、做事流程与方法、逻辑思维与分析等，都是通用的，可以迁移到新的领域，而它们在你个人能力中的占比，可能超过70%！

当你想要进入新领域时，会不自觉地被“我已经浪费了那么多时间，必须快速达到某个高度才成”这种想法困住，一旦你把焦点放在了“速成”上，就会发现“哇好难啊，不可能学会”，就会产生“恐怕很长时间内我这个职场老手要被人当作小白来鄙视”这种想法，就会望而却步。这样一来，进入新领域的难度，就被“在别处花费的很多时间”和“在新领域崛起必须要很短时间”这种强烈对比给放大了。但实际上，你现在熟悉的领域，不也是从零开始花了很多时间一步一步走过来的吗？

关键是要有一种学习型心态，要能够归零、空杯，要相信一切皆有可能。

2.薪水落差

程序员的薪水远高于其他行业，很多朋友都是因为这一点选择软件开发，或者因为这一点坚持做下去。

当我们没有别的追求时，追求金钱是一个不错的选择。但这话反过来就成了软件开发者的魔咒：当我们发现了自己的追求，想要离开时，我们已经获得的薪酬水平就会成为我们的羁绊。

干过几年开发工作的朋友，想转行，薪水都会经历一个断崖式下坠，这是多数人不能承受的，也是人的天性（损失厌恶）——得到了就不想再失去。

我们习惯性地认为职场只能一个台阶一个台阶地往上走，我们习惯性地认为薪水只能一年比一年高，我们不能接受成长过程中的凹陷。

如果我们能回到“为什么工作”这个元问题，这样的不舍和纠结就会瞬间散去。我们工作，我们要不断攀升的薪水福利职位，一方面是为了让我们的生活更安全，更有幸福感、意义感，另一方面是为了证明自己有价值，实现自我成就。那么当我们发现了一件直接就能带给我们意义感、价值感的事情时，去做它不是更好吗？你不仅不用拿钱去买、去寻找，而且在做它时还能赚到钱，这不是更划算、更有价值的选择吗？

3.缺乏技能

程序员的技能树围绕着软件开发构建，需求分析、设计、编码、调试、配置环境……一旦你想要转战他方，就会发现，好像自己这么多年就只会开发软件了，别的什么都不会，技能严重缺乏，然后你可能就会缩回去，算了，还是先做开发吧。

介绍路径依赖时已经提到过这一点。

但实际上，人的能力分为专业能力和通用能力。

专业能力是指只能在某个专业领域发挥作用的能力，比如“使用Java编程”这种技能，在软件开发领域有用，进入房地产零售方向，就没什么价值。专业能力可以通过培训、练习、做项目这种套路快速习得。

通用能力指的是可在不同领域间迁移的能力，比如学习、演讲、组织、沟通、计划、管理、讲授、分析、英语、PPT制作与展示等能力，在软件开发领域有用，拿到餐饮行业一样有用。

作为开发者，在最初的2~3年，花在专业能力上的时间会多于通用能力，你会发现Java用得越熟练，MyBatis玩得越顺畅，你的工作成就越好。

但如果你一直将焦点放在专项能力的淬炼上，忽略沟通、协调、管理、讲授、分析等通用能力，久而久之，你就会发现自己在工作上很难再有大的突破，会陷入“明明我技术很厉害可是却无法获得与其匹配的工作成果”这种怪圈。

这往往是因为你忽略了一点：**所有问题，最终都是人的问题。**而要解决人和人之间的问题，专项技术能力很难派上用场，你必须得综合运用你的各种通用能力，才能给自己打造一个和谐的工作环境。有了和谐的工作环境，你才可能用你的专业能力来解决问题。

从这个角度看，我们在工作中，要培养专业能力，更要抓住机会练习，打磨通用能力。这样当你切换领域时，就可以把通用能力全部带走，并且在通用能力的保驾护航下，通过培训、练习、做项目等方式迅速提升专业能力，让自己在较短的时间内融入新的环境，胜任新的工作。

4.他人的期待

别人会对你有期待，他们希望看得懂你，不希望你带给他们看不懂的意外。我们给身边的每一个人、给我们可能看到的每一个人都编制了一个“人设”，认为他们就该按照我们设定的角色去发展，如果他超出了我们预想的“人设”，我们就会诧异、不解、愤怒，进而否定、打压。这就是我们的生活中贴标签盛行不衰的原因。

就这一点而言，父母、伴侣、朋友对我们的期望，尤其会给我们带来巨大的压力，父母会不希望你放弃稳定又高薪的工作，伴侣会担忧你的选择不能保障优裕的家庭生活，朋友会觉得你不应该瞎折腾让自己越来越不值钱……

除了亲戚朋友，但凡稍稍熟悉你的人，都会对你有期待、有评价，都习惯用他们感到舒服的视角来框你。

别人会因为你做了软件开发工作，就把你等同于程序员这一身份，会一直强化你的优势——技术，而不看你别的方面。

比如我参加智空间的咖访谈，访谈结束后，有好几位朋友找我，就是因为我技术好，就要我帮忙做技术方案或者到他们公司从事技术工作。

比如我参加在行西安的线下聚会，有的朋友知道我是程序员，就准备外包项目给我。

别人看到的，都是你过去走过的路。

当你因为别人给你贴的标签而举步不前时，要想想：

你不是你的工作，你不是你的技术，你不是你的成功，你不是你的失败，你是一个立体的、多元化的人，你具有多重角色和身份，你在开发软件时是程序员，你在写作时是作家，你在讲课时是老师，你

在照顾孩子时是爸爸妈妈，你在陪伴父母时是孩子，所有的角色都是你，但没有一个单一的角色代表全部的你。

这里的思维转变点在于：

- 你不是你的工作
- 你不是你的标签
- 你不是别人对你的评价
- 别人的评价不会改变已然发生的事实

意识到这些，再进一步，你就能看到：你的价值来源于你能做什么事情、能解决什么问题，你是因为做了什么事情而成为了你，而不是别人的评价——那是后发的，是你做了事情之后他们才走的马后炮。

所以，Go your own way; let others talk！

5.对不确定性的抗拒

开发者的工作，有相当一部分时间在与计算机等设备打交道，按一定的规则输入，得到预期中的输出，这样的方式，有很强的一致性、确定性、稳定性。习惯之后，就比较排斥变化，抗拒不确定性。

你喜欢你写的APP不定时闪退吗？

你喜欢你负责后端服务隔三差五崩溃吗？

你喜欢产品经理一天改三次需求吗？

不喜欢！

你习惯并喜欢确定、稳定、可预期！

你不愿意面对不确定性，不愿意承受波动。

但转型充满了不确定性，你可能会选错方向，也可能最初的方向正确但做上一段时间发现不喜欢，还可能做了几年也没什么起色，所以开发者在不能确定转型100%会成功时，往往很难迈出第一步。

同样的行为产生同样的结果，这是软件开发中的一致性。这种一致性，在人身上也是一样的。你一直做一件事情，就一直会得到同一个结果。你一直做，反复做，做到轻车熟路，还一直做，还是那个结果，你不愿意尝试其他的事情，那么你就不会有大的变化，就会一直在老路上徘徊，很难有新的发展。

要想让工作和生活走出枯井无波的状态，就要引入一些变化，承受这些变化发生过程中伴生的不确定性。你不断给自己尝试的机会，就能不断拓展自己的认知和能力边界，就能不断发现自我，就极有可能找到自己的天赋，发展出新的事业。

8.3 转型的分类

我们通常所说的“工作”，有行业和职能两个要素。

所谓行业，是由某一类经济活动性质相同或相近的组织所构成的。比如少儿英语培训、游戏、医疗器械、操作系统、通信、出版等，都算是行业。招聘网站是你了解行业划分的好地方，图8-1所示是我从智联招聘（http://www.zhaopin.com/）的首页，搜索职位那里截取的。

这里要注意的是，开发者的行业属性有两重：计算机软件和所做软件所属行业。举个例子，猿题库的开发者，就有计算机软件和教育两个行业属性，而且，往往与你所做软件相关的行业属性更重要。

所谓职能，指的是某个职位所承担的任务、作用和功能。比如软件开发工程师和研发经理，就是两种不同的职能（所以开发者升任研发经理，不是职位序列自然发展的结果，而是一种转型）。

我们把职能分为5类：管理、专业技术、自由职业、创业、投资。投资其实是一种综合性职能，它有管理、专业技术、创业几种特性。我们在这一节讨论转型分类时，暂不考虑它。



图8-1

以职能为纵轴，行业为横轴，可以绘制一张转型分类图，如图8-2所示。

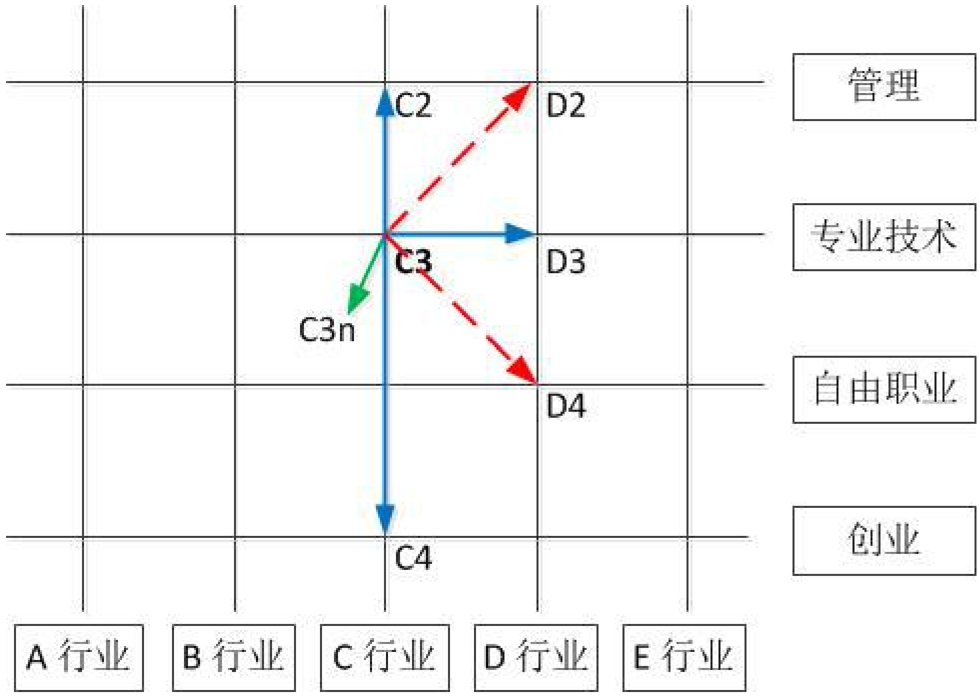


图8-2

假定当前职业为C3，那么“跳槽”就有4种模式：

1）C3→C3n，同行业内换一家公司原职能方向发展，比如你从酷派的Android Framework工程师跳到华为去做Android Framework工程师，就属于这一种。

2）C3→C2或C3→C4，行业不换，职能转变。比如一个iOS应用开发工程师因为工作出色被任命为研发经理，就属于这一种。

3）C3→D3，职能不变，行业转换。比如一位程序员原来在通信行业做网上营业厅的开发，现在转到教育行业做直播教学类的软件，就属于这一种。

4）C3→D2或C3→D4，行业、职能都转换。比如有位朋友原来在运输行业开长途汽车，后来通过自学进入一家油服公司做测井软件开发，就属于这一种；再比如有朋友在彩票行业做APP开发，搞累了回家包了100亩地种果树，也属于这一种。

第2）、3）、4）是三种常见的转型模式。第1）种，就是我们通常意义上的跳槽，算不上是转型。

第4）种是最难的转型方式，因为行业和职能都不相关。但如果你发现了自己的天赋，并且有足够的准备和执行力，一样可以做到。

徐沪生毕业后从事软件开发工作，两年存款15万元，却在2014年突然放弃了稳定的高薪工作，专攻写作，两个月后出版《少年啊，前路漫漫》，转身为励志作家。他的转型，就是第4）种。

徐沪生是少数派，大部分人很难在行业、职业都不相关的情况下去转型，那需要非常大的勇气。但如果你转型的方向，和你现在的职业真的没有任何相关性，怎么办呢？

此时可以考虑两步走策略，每步转换使用一种相关性。比如一位ERP方向的开发者，想成为英语培训行业的研发经理，可以先用Java作为相关性，寻找英语培训行业的开发岗位，胜任后，再在公司内谋求转型为研发经理的机会。

8.4 与开发者相关的转型方向

转型时，在行业或职能上有一个维度相关，就会相对容易一些。程序员转型也可以考虑这种模式。

比如你只是不想做开发了，还想做和软件相关的事情，就可以考虑如下方向：

1）项目经理

2）产品经理

3）需求分析人员

4）文档开发人员

5）测试人员

6）软件售前人员

7）软件售后人员

8）讲师

9）技术作家

10）运维人员

11）DBA

12）配置管理人员

13）文档开发人员

14）面向开发者的猎头

15）版本工程师（Software Build Engineer）

16）质量管理人员

17）软件实施人员

8.5 如何确认哪个职业适合你

当你不想再做开发，或者被家人劝退，或者被辞退、不得不寻找转型方向、却又没有明确的方向时，可以使用本节提供的职业定位方法——**人事物模型，从喜欢的人、事、物中找到适合你的职业。**

遵循下面5个步骤，就可以快速找到你想做的职业：

1）罗列你感兴趣的或印象深刻的人、事、物

2）分析它们可以关联到的职业，形成清单

3）对清单中的职业做信息搜集分析，筛选出你感兴趣的职业

4）选择某个职业，学习相关知识、技能

5）业余时间做做看，或者找兼职、实习机会做做看

图8-3表示了这个流程。

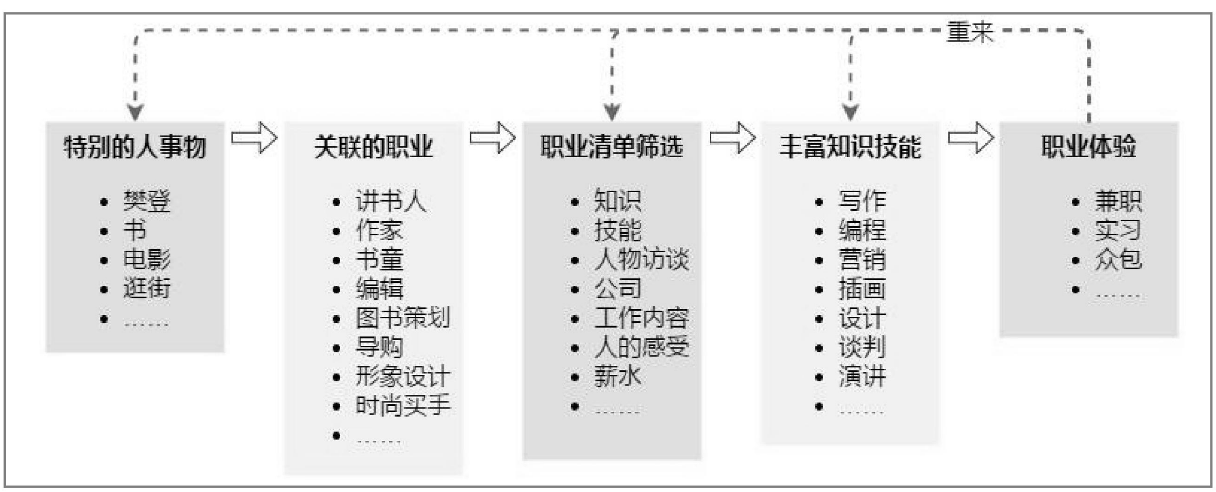


图8-3

1.那些特别的人、事、物

我们都有感兴趣的人，一定有。比如我，对路遥、王小波、侯捷、杰拉尔德·温伯格、樊登、邹鑫等很感兴趣。你呢？

我们都有很多感兴趣的事情，或者可以很投入地去做的事情。比如我对读书、买书、写书、荐书、研发课程就很感兴趣。再比如有的人就很喜欢打游戏，能连续三四天不睡觉地打游戏。还有的人喜欢看电影，院线上映的电影，一场都不会落下。

我们还可能做过一些让我们有成就感的事情。比如你自己从西安骑行到北京，比如我上大学时拿《中国青年》的稿费……

我们也有很多感兴趣的物品，比如有的人喜欢邮票，有的人喜欢各种各样的ThinkPad笔记本，有的人喜欢翡翠，有的人喜欢衣服，有的人喜欢书……

你可以尽情回忆、梳理自己，把这些都列出来。

2.从人、事、物导出职业清单

你感兴趣的人、事、物，多数都可以关联到某种职业上来。

比如我喜欢路遥，对应作家这种职业；比如我喜欢侯捷，对应到技术作家、软件开发工程师、讲师；比如我喜欢樊登，对应到讲书人这种职业……

比如我爱荐书，可以关联到拆书家、讲书人、图书销售员、图书管理员、书童等职业；比如你喜欢电影，可以关联到影评人、放映员、检票员、院线值班经理、编剧、导演、演员、摄像、动画制作、剪辑、灯光、配音演员、翻译等等职业……

比如喜欢邮票，可以做邮票收藏与鉴赏类的工作，或者邮票设计师；比如喜欢衣服，可能关联到服装设计师、服装销售、服装搭配师、导购等职业……

你可以通过下面三种方式找到与你的人、事、物清单相关的职业：

1）从人、事、物导出知识、技能，根据知识、技能到招聘网站检索

2）搜索引擎

3）请身边的人说出三个与你了解的人、事、物相关的职业

可能这些还是太笼统，那么下面的建议也可以参考。

【人】

从你感兴趣的人，分析出关联的职业，这些职业，就可能有你想做的。这也是我们都熟悉的：榜样的力量。

分析与某人相关的职业，可以考虑4个方面：他做过的职业、他做的事情、他的身份标签、他的生活状态。

如图8-4所示是我分析樊登读书会创始人樊登得出来的，可以参考。

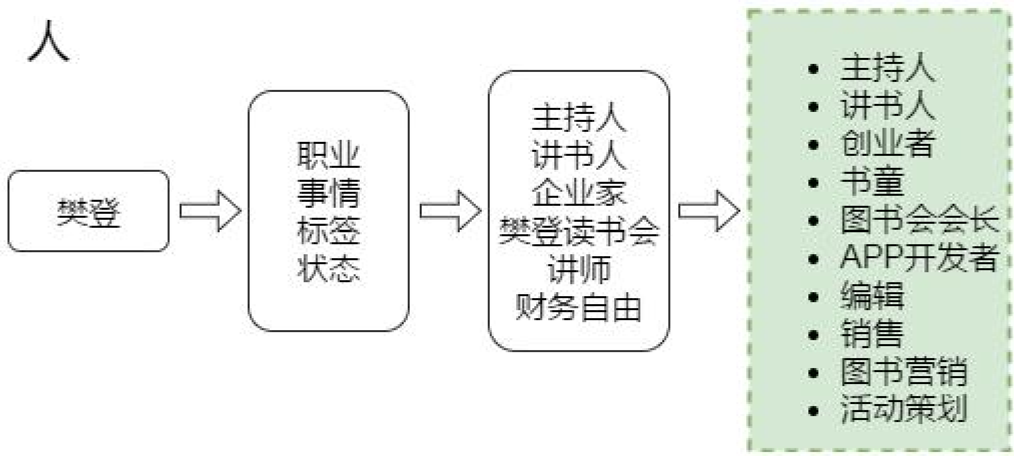


图8-4

【事】

你喜欢做的事情里，关联了很多职业。只是我们平常做什么事，做了就算了，习惯了，一般不去琢磨这件事都牵涉到什么环节、用了什么知识什么技能、与什么人、什么职业有关联，如果你仔细去琢磨，像打游戏、打牌、看电影、读书、逛街、旅游等你喜欢做的事情里，都有很多职业。

下面这些维度，可以帮助你去分析一件你喜欢做的事情，推导出一个职业清单：

a）事情是什么

b）所需物料

c）所需场地

d）运营者是谁

e）别人怎么做

f）谁以此赚钱

g）谁与此相关

我们以打游戏这件事来分析一下，看我贴的图，如图8-5所示。

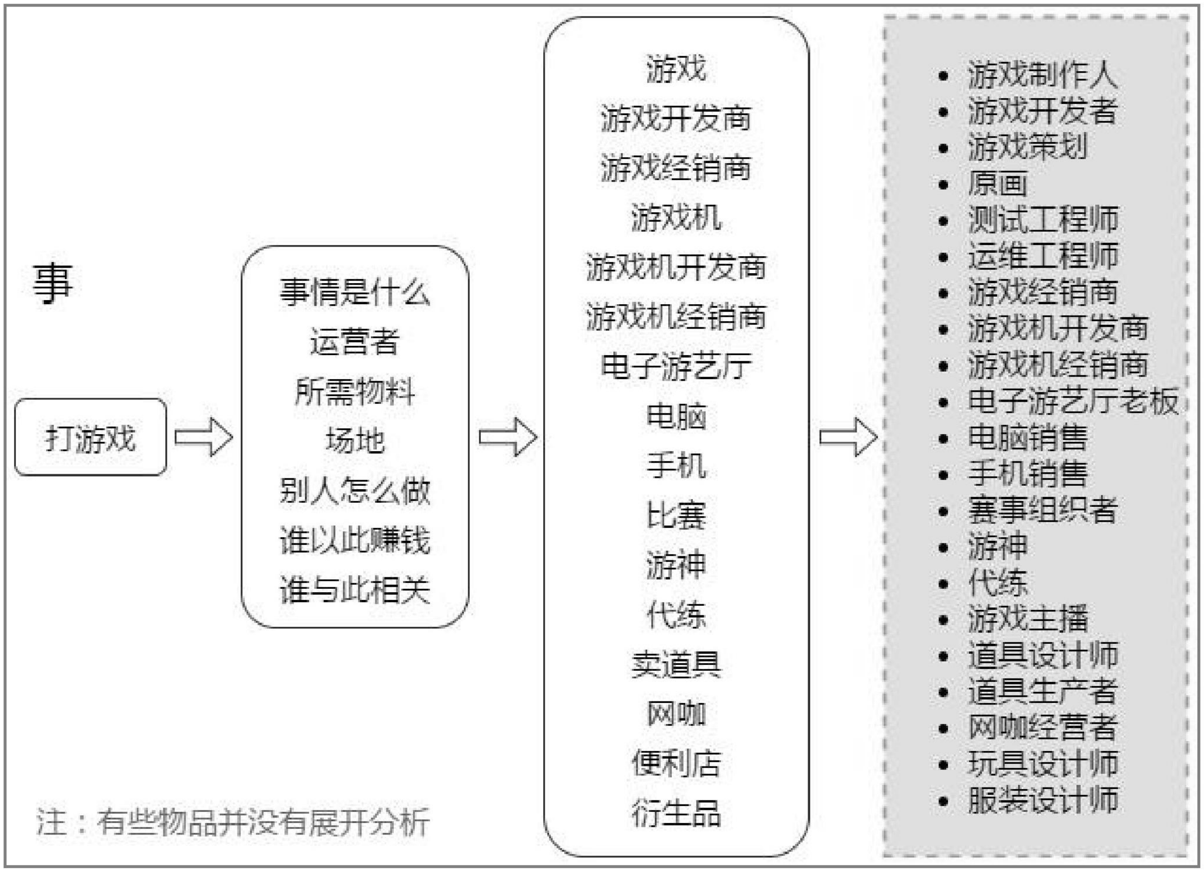


图8-5

【物】

对于物品，可以考虑：这些物品是如何制造出来的、它们怎样被使用、怎样抵达用户手中、与它们相关的有什么场景。

从这些角度出发，你的思维就会被打开，就能看到更多的可能性。

如图8-6所示的这张图描述了寻找和书相关的职业的过程。

我们把探索职业方向的过程分成两步：第一步先列出可能感兴趣的职业；第二步再评估能否做到。这样做分离了想要什么和能否实现这两个过程。

所以我们在做关联职业搜集时一定要注意：千万不要因为自己当下看起来做不到某个职业而忽略掉它，在这个阶段，不要想自己是不是能做到，只去想某个人、事、物可能关联什么职业。这是一个头脑风暴的阶段。

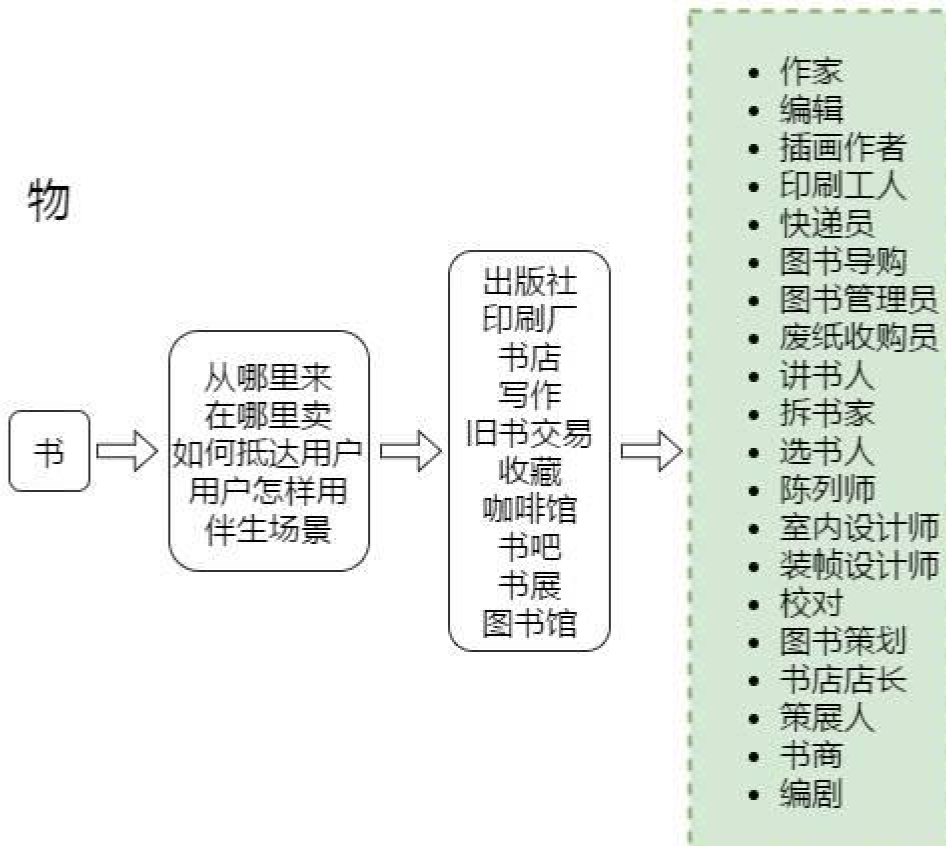


图8-6

3.职业清单筛选

现在，我们有了一份职业清单，这个清单可能很长，上面可能有几十或者几百种职业。这么多职业，必须得进行筛选，找出我们确实感兴趣的。

根据两个原则来筛选：

1）相关性原则（行业相关或者知识技能相关）

2）想要指数

先做一个类似下面的表格，把上一步的职业清单中的职业导入：

职　业	相 关 性	想要指数	综合得分
作家			
职业规划师			
主持人			
自由开发者			
IT 咨询师			
……	……	……	

现在我们来看两个原则。

所谓“相关性”，指目标职业和你现在的个人商业价值在某方面有关联，比如知识技能，比如行业或业务，比如人脉。当你与某个职业具备相关性时，切换过去的难度就相对小一些。

这是我们在转型时可以考虑的、比较现实的策略：通过把转型分解为多个阶段，让每个阶段的目标都和当下的个人商业价值有相关性，降低转型难度。

那么怎样判断相关性呢？需要做两方面的数据搜集与分析工作：

1）我拥有的商业价值，包括知识、技能、经历、人脉、天赋5方面

2）目标职业需要什么知识、技能、经历、人脉

这部分在第6章有详细介绍。

目标职业需要的知识、技能、经历、人脉，可以通过到招聘网站搜索相关职位来搜集，也可以通过搜索引擎搜索，或者到知乎上提问、搜索，还可以利用在行、分答、值乎等平台，找到从事对应职业的人来了解。这是一个说起来简单做起来烦琐、耗时的工作，需要有一些耐心。

有些朋友可能会觉得转型时根据相关性来选择职业的过程比较漫长，受不了，就想一步到位。此时可以用“想要指数”来进行选择或筛选。

所谓“想要指数”，指的是你想做某种职业的强烈程度。

比如你感到自己如果现在不去做，这辈子都无法安心，不能原谅自己，那么想要指数就很高；比如你觉得这事放上十年八年再做也没什么关系，那么想要指数就趋近于0了。

怎么确定自己的想要指数呢？有两种方法：

1）想象未来三年自己的样子（你的形象、你在做什么事情、你和什么人在一起、你过着什么样的生活），或者更远，这辈子，你最终想成为什么样子。

2）如果不做这件事，自己会有多痛苦。痛苦越强烈，对应的想要指数就越高。

有了想要指数和相关指数之后，把它们量化一下，比如最高分取10分，最低分取0分，填入前面的表格中，例如：

职　　业	相关性	想要指数	综合得分
作家	3	10	13
职业规划师	6	8	14
主持人	1	5	6
自由开发者	8	5	13
IT 咨询师	7	7	14
……	……	……	

有了这张表，你就可以进行筛选了。按照下面的策略来筛选，可能好一些：

- 保留相关性得分大于等于3的职业

- 保留想要指数大于等于5的职业

做了上面两种筛选之后，可以再考虑综合得分，比如：保留综合得分大于等于12的职业。

这样做过两轮之后，你的表格中，应该就会只有少数几个职业了。

如果此时你的表格中还有很多项，说明你对相关性和想要指数的判断标准比较宽松，可以再来一轮，把标准放严一些。

做过几轮筛选之后，把你的职业表格中的项数，控制在5项以内（选择太多会很纠结）。

现在，我们需要在剩下的5个职业中选出一个来开始下一步行动了。

选起来好纠结……

有时候我会用点兵点将的方法，或者扔硬币、抓阄等。有时这也是不错的方法——当你真的无法取舍时，交给所谓的天意吧。

不过作为理性的人，我更愿意使用下面这个办法：

在职业的若干特征中，选出最看重的三个，根据这三个特征来选择，哪个职业符合的特征多，就选哪个。（请参考第5章“跳槽8问”中5.2节下面的“需求供给分析法”部分）。

有人说这样选择似乎还是纠结啊……

好吧，我们再来一个最简单易行的办法吧！

问自己一个问题：在职业上，最看重的因素是什么？

注意，只选一个因素。当你只用一个要素来选择职业时，选择就变成了判断，Yes or No，就很简单了。

这一点，我特别喜欢《反脆弱》中的观点：

如果你做某事的理由超过一个，那就不要做。这并不意味着一个原因比两个原因更好，只是说，通过努力想出一个以上的原因，你实际上正在试图说服自己做一些事情。显而易见的决定（在错误面前是强韧的）不需要一个以上的原因。

如果你始终不能决断，也可以约我面对面一对一深聊，扫描图8-7所示的二维码，即可通过在行约我。

图8-7

4.丰富相关的知识和技能

终于过了最纠结的时刻，现在，我们知道最想做的那个职业是什么了，接下来就该为它做准备了。

准备工作可以按先后顺序分为下列三步：

1）了解这个职业具体做什么，即工作内容

2）需要什么知识、技能

3）储备相应的知识、技能

1）、2）两点设计的信息，可以通过招聘信息、搜索引擎、问答平台（在行、分答、值乎、微博问答、知乎Live）、社交渠道（微信群、QQ群、朋友圈、人脉）等获取到。接下来我们主要看如何储备目标职业所需的知识和技能。

有下列方式：

1）购买相关书籍，阅读，练习

2）参与相关在线课程（一般有免费也有收费的）、实战营、训练营等

3）线下培训

4）私教

个人比较推荐按照上面的顺序来尝试，先自主学学看，因为不论你转换到哪个职业，独立学习和解决问题的能力都是非常重要的。如果你不能自己获取自己想要的信息，不能自己学习工作中需要的知识、技能，基本上很难较好地生存下去。

图书和在线课都是低成本的，几十元、几百元、几千元就可以让你迅速了解一个领域。比如我2017年就参加了方军老师的3期知识经济实战营，学习了如何设计知识产品。

当你自主学习一段时间后，对目标职业所需知识、技能有了更深入的了解，想要加速，或者想要一个共同学习促进的环境，可以考虑线下的训练营或者培训机构。

现在各类职业培训机构应有尽有，驾校、会计、心理咨询师、软件工程师、营养师、药剂师、护士、教师、美容师、挖掘机、职业规划师、精算师、金融分析师、平面设计、插画、漫画……

只要是你想做的，基本上都能找到对口的培训机构。

2015年我参加了向阳生涯为期7天的线下培训，一群喜欢职业规划的小伙伴每天花十几个小时一起学习、讨论、实战训练，氛围非常好，对于我迅速掌握职业规划技能体系有很大的帮助。

对于找不到培训机构的职业，或者如果不满足大班培训，也可以再寻找私教，比如到在行上约具有相应履历的人，或者寻找培训机构

的教师，或者联系知乎大V……有很多渠道，只要你想要，就可以找到。

上面这些事情，可以在业余时间做、请假做或者离职专门做，采用哪种方式，主要看你的经济状况和个人对某种方式的内心看法。

5.做做看

当你具备一定的知识、技能后，就需要找机会做做看了。

实际的职业体验非常重要，你不亲身去做，永远不知道这个职业是不是真的适合你。这就像买鞋子，你的脚再标准，鞋码再标准，穿上都可能难受，必须得上脚试，还必须得走上一阵子，才能初步判断出来。

通常有这些方式可以尝试：

- 利用工作中的机会来锻炼
- 兼职
- 实习
- 网络上的众包
- 独立提供服务

先说工作中的机会。比如你是软件开发工程师，想转UI设计，那么你自己学习绘画、AxurePro、PS等技能后，就可以自己制作原型图、效果图、切图等，就能够锻炼。

再来看兼职，机会更多，平台更多。比如程序员客栈、兼职猫、猪八戒、在行等线上平台，有很多兼职可做的事情。线下也有很多机

会，如果你留意，就可以看到商场、超市等人流密集的地方，经常有招聘兼职人员的信息，还有当地的人才市场，有很多兼职机会。

实习的机会对于学生来讲比较多，像51job就有实习频道，智联招聘也有很多实习信息。对于已经工作几年的朋友，实习可能会演变成兼职的方式，或者在你以增加实践经验和职业体验为主要目的时，也可以考虑先做实习生。

再聊聊众包，现在网络上非常多。比如软件开发，就有很多众包的机会，程序员客栈、码市、猪八戒等平台，都有；比如猎头，现在也有一些人通过众包的方式来找人，你如果接受了众包任务，就可以获得分成；再比如翻译，也有众包的方式。

独立提供服务对于具备某些技能的朋友很有用。比如我女儿上幼儿园时，班上有个孩子的妈妈，提供美甲、修眉、化妆等服务，很多朋友找这个妈妈做；比如你插画画得不错，完全可以在微博上加入“求约稿”来接商稿；比如你文案写得好，可以对外提供产品文案写作……机会永远比你想象到的多，关键是你要先具备某种技能。

8.6 转型的最佳实践

如果你真的要转型，按照下列步骤推进是比较现实的：

1）找到转型方向。

a）与软件开发相关的方向；

b）通过人事物模型寻找新的职业目标。

2）储备目标职业所需知识、技能。

3）找机会实践，体验，看自己是否还愿意继续做，如果不愿意，回到1）。

4）在目标工作地区，寻找感兴趣的公司和职位。

5）针对每一个公司的每一个职位，准备一份简历，突出你与目标职位匹配的知识、技能、经历。

6）优先考虑通过人脉、内推等方式获取面试机会；实在找不到人推荐，再考虑网络、现场招聘等方式。

重复上面的过程，迭代，优化，直到找到工作。

（在这里推荐求职史上最强的指南：《你的降落伞是什么颜色》）

在转型时有两种选择：

1）离职，全力转型。

2）一边继续当前的工作一边利用业余时间转型。

当你有经济压力或者担心转型失败后很难再找到类似当下的工作时，可以考虑采用第2）种方式，也就是双职业策略。

双职业策略

《小强升职记》的作者邹鑫，原本在一家国企做开发，不喜欢那种工作氛围，干得不是很开心。后来对时间管理产生兴趣，就主动学习，分享，慢慢通过写作博客建立影响力，后来出版了《小强升职记》，在时间管理领域建立了个人品牌，收入超越了本职工作后，他选择了离职，转型为时间管理自由讲师和作家。

邹鑫采用的转型策略，就是双职业策略。

所谓双职业策略是指：保持当前工作，业余时间丰富目标职业所需知识、技能，尝试用新技能变现，当新技能带来的收益超过生存线或者新技能可以帮助你找到新的工作时，再切换到目标职业。

图8-8可以很好地说明这种策略。

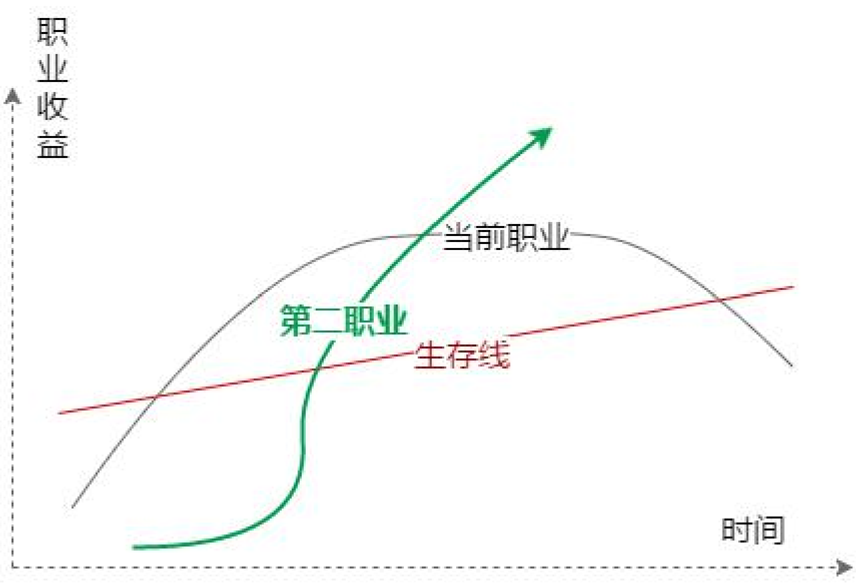


图8-8

双职业策略要求你能够很好地管理你个人的时间和精力，在双职业并存的那段时间，会有比较大的挑战。

2017年上半年，我一边做软件开发，一边在业余时间通过知乎Live、CSDN直播、StuQ直播等做线上分享，分享所需要的素材，只能在上班前、下班后、周末这些时间段准备，非常忙碌，我几乎没有时间关注爱人和女儿。几个月的重叠期，给我和家庭带来了非常大的压力。

所以如果你是单身，利用双职业策略转型，可能压力会小一些，如果你有家庭、有孩子，采用这种方式，就要征得他们的同意，妥善安排时间和精力，并设定一个期限。

附录A 私房书单

这里推荐的图书，是我读过并认为不错的。分成如下5类：

- 开发者素养
- 技术与管理
- 向上管理
- 自我成长与规划
- 财富思维

这里只列书名，感兴趣的，可以搜索书名去了解。另外我的订阅号“程序视界”，每周会采用抽取精华的方式推荐一本书，可以关注。微信扫描下面的二维码：

1.开发者素养

- 《代码大全》（史蒂夫·迈克康奈尔）
- 《重构：改善既有代码的设计》（福勒）
- 《人月神话》（FrederickP.Brooks.Jr.）
- 《程序开发心理学》（杰拉尔德·温伯格）
- 《颠覆完美软件：软件测试必须知道的几件事》（杰拉尔德·温伯格）
- 《程序员修炼之道：从小工道专家》（Andrew Hunt，David Thomas）
- 《高效程序员的45个习惯，敏捷开发修炼之道》（Venkat Subramaniam，Andy Hunt）
- 《高效能程序员的修炼》（Jeff Atwood）
- 《软技能：代码之外的生存之道》（John Sonmez）
- 《程序员的职业素养》（Robert C.Martin）
- 《程序员的思维修炼》（Andy Hunt）
- 《人件》（Tom DeMarco，Tim Lister）
- 《计算机程序设计艺术》（Donald E. Knuth）
- 《软件工程：实践者的研究方法》（普雷斯曼）
- 《聊聊架构》（王概凯）
- 《大型网站技术架构：核心原理与案例分析》（李智慧）

- 《你好哇，程序员》（安晓辉）
- 《逻辑思考力》（西村克己）
- 《逻辑思维简易入门》（加里·西伊，苏珊娜·努切泰利）
- 《学会提问》（尼尔·布朗，斯图尔特·基利）
- 《你的灯亮着吗？》（杰拉尔德·温伯格）
- 《系统化思维导论》（杰拉尔德·温伯格）
- 《系统之美》（德内拉·梅多斯）

2.技术与管理

- 《人月神话》（FrederickP.Brooks.Jr.）
- 《人件》（Tom DeMarco，Tim Lister）
- 《成为技术领导者》（杰拉尔德·温伯格）
- 《软件工程：实践者的研究方法》（普雷斯曼）
- 《上任第一年1》（琳达·希尔）
- 《横向领导力》（罗杰·费希尔，艾伦·夏普）
- 《别让猴子跳回背上》（威廉·安肯三世）
- 《项目百态，深入理解软件项目行为模式》（Tom DeMarco、Peter Hruschka、Tim Lister、Steve McMenamin、James Robertson、Suzanne Robertson）
- 《项目经理应该知道的97件事》（Barbee Davis）

- 《项目管理修炼之道》（Johanna Rothman）

- 《目标管理实务手册》（串田武则）

- 《所谓情商高就是会说话》（佐佐木圭一）

- 《高绩效教练》（约翰·惠特默）

- 《内向者沟通圣经》（珍妮弗·康维勒）

- 《内向也是一种优势》（希尔维娅·吕肯）

- 《卓有成效的管理者》（彼得·德鲁克）

- 《管理的实践》（彼得·德鲁克）

- 《带人的技术：不懂带人你就自己做到死！》（石田淳）

- 《影响力》（罗伯特·B·西奥迪尼）

- 《领导梯队：全面打造领导力驱动型公司》（拉姆·查兰，斯蒂芬·德罗特，詹姆斯·诺埃尔）

- 《系统思考》（丹尼斯·舍伍德）

3.向上管理

- 《如何管理你的老板》（杰伊）

- 《上任第一年2》（琳达·希尔，洛厄尔·肯特·莱恩巴克）

- 《向上管理，做高效能下属》（李卓汐，秦浩洋）

- 《向上管理的技术》（門脇龍一）

- 《向上汇报》（弗雷德里克·吉伯特）
- 《卓有成效的管理者》（彼得·德鲁克）
- 《管理的常识》（陈春花）

4.自我成长与规划

- 《少有人走的路》系列4本（斯科特·派克，托马斯·摩尔）
- 《高效能人士的七个习惯》（史蒂芬·柯维）
- 《发现你的天赋》（肯·罗宾逊）
- 《A Life at Work（这辈子，我最想做的事）》（托马斯·摩尔）
- 《自卑与超越》（阿尔弗雷德·阿德勒）
- 《你要如何衡量你的人生》（克莱顿•克里斯坦森，詹姆斯•奥沃）
- 《你的生命有什么可能》（古典）
- 《你的降落伞是什么颜色》（理查德·尼尔森·鲍利斯）
- 《高效演讲》（彼得·迈尔斯）
- 《干法》（稻盛和夫）
- 《金字塔原理》（巴巴拉·明托）
- 《做事的常识》（小仓广）
- 《刻意练习》（安德斯·艾利克森，罗伯特·普尔）

- 《精要主义：如何应对拥挤不堪的工作与生活》（格雷戈·麦吉沃恩）
- 《现在，发现你的优势》（白金汉）
- 《现在，发挥你的优势》（白金汉）
- 《关键对话》（科里·帕特森约瑟夫·格雷尼，罗恩·麦克米兰，艾尔·史威茨勒）
- 《非暴力沟通》（马歇尔·卢森堡）
- 《反脆弱》（纳西姆·尼古拉斯·塔勒布）
- 《10天谋定好前途》（洪向阳）
- 《当时忍住就好了》（肯·林德纳）
- 《持续的幸福》（马丁·塞利格曼）
- 《放弃的艺术》（佩格·斯特里普，艾伦·伯恩斯坦）
- 《拆掉思维里的墙》（古典）
- 《精进：如何成为一个很厉害的人》（采铜）
- 《小强升职记》（邹鑫）
- 《如何阅读一本书》（莫提默·J. 艾德勒，查尔斯·范多伦）
- 《这样读书就够了》（赵周）
- 《好好学习：个人知识管理精进指南》（成甲）

- 《复盘：对过去的事情做思维演练》（陈中）
- 《你的知识需要管理》（田志刚）
- 《定位》（艾·里斯，杰克·特劳特）
- 《22条商规》（艾·里斯，杰克·特劳特）

5.财富思维

- 《富爸爸穷爸爸》系列（罗伯特·清崎、莱希特）
- 《小狗钱钱》系列（博多·舍费尔）
- 《思考致富》（拿破仑·希尔）
- 《富足，从心开始》（吕迪格·达尔克）
- 《稀缺：我们是如何陷入贫穷与忙碌的》（塞德希尔·穆来纳森，埃尔德·沙菲尔）
- 《富足：改变人类未来的4大量》（彼得·戴曼迪斯，史蒂芬·科特勒）
- 《财报就像一本故事书》（刘顺仁）
- 《大家的财税学》（李炜光）